Current Issues in Linguistic Theory

Recent Advances in Natural Language Processing III

EDITED BY
Nicolas Nicolov
Kalina Bontcheva
Galia Angelova
Ruslan Mitkov

RECENT ADVANCES IN NATURAL LANGUAGE PROCESSING III

AMSTERDAM STUDIES IN THE THEORY AND HISTORY OF LINGUISTIC SCIENCE

General Editor

E.F. KONRAD KOERNER
(Zentrum für Allgemeine Sprachwissenschaft, Typologie und Universalienforschung, Berlin)

Series IV – CURRENT ISSUES IN LINGUISTIC THEORY

Advisory Editorial Board

Lyle Campbell (Salt Lake City); Sheila Embleton (Toronto)
Brian D. Joseph (Columbus, Ohio); John E. Joseph (Edinburgh)
Manfred Krifka (Berlin); E. Wyn Roberts (Vancouver, B.C.)
Joseph C. Salmons (Madison, Wis.); Hans-Jürgen Sasse (Köln)

Volume 260

Nicolas Nicolov, Kalina Bontcheva, Galia Angelova and Ruslan Mitkov (eds.)

Recent Advances in Natural Language Processing III.
Selected papers from RANLP 2003.

RECENT ADVANCES IN NATURAL LANGUAGE PROCESSING III

SELECTED PAPERS FROM RANLP 2003

Edited by

NICOLAS NICOLOV
IBM T.J. Watson Research Center
KALINA BONTCHEVA
University of Sheffield
GALIA ANGELOVA
Bulgarian Academy of Sciences
RUSLAN MITKOV
University of Wolverhampton

JOHN BENJAMINS PUBLISHING COMPANY
AMSTERDAM/PHILADELPHIA

The paper used in this publication meets the minimum requirements of American National Standard for Information Sciences — Permanence of Paper for Printed Library Materials, ANSI Z39.48-1984.

Library of Congress Cataloging-in-Publication Data

RANLP 2003 (2003 : Samokov, Bulgaria)
Recent advances in natural language processing III : selected papers from RANLP 2003 / edited by Nicolas Nicolov … [et al.].
p. cm. -- (Amsterdam studies in the theory and history of linguistic science. Series IV, Current issues in linguistic theory, ISSN 0304-0763 ; v. 260)
Papers from the RANLP conference held Sept. 10–12, 2003 in Samokov, Bulgaria.
Includes bibliographical references and index.
Computational linguistics--Congresses.
P98.R36 2003
410/.285--dc22 2004062362
ISBN 90 272 4774 9 (Eur.) / 1 58811 618 2 (US) (Hb; alk. paper)

John Benjamins Publishing Co. • P.O.Box 36224 • 1020 ME Amsterdam • The Netherlands
John Benjamins North America • P.O.Box 27519 • Philadelphia PA 19118-0519 • USA

Contents

III. TAGGING, PARSING AND SYNTAX

IV. INFORMATION EXTRACTION

V. TEXT SUMMARISATION AND DOCUMENT PROCESSING

VI. OTHER NLP TOPICS

Editors' Foreword

This volume brings together revised versions of a selection of papers presented at the International Conference on "Recent Advances in Natural Language Processing" (RANLP 2003) held in Samokov, Bulgaria, 10–12 September 2003. The aim of the conference was to give researchers the opportunity to present new results in Natural Language Processing (NLP) based on modern theories and methodologies.

The conference was preceded by three days of tutorials. Invited lecturers were:

Piek Vossen (*Irion Technologies*)
Wordnet, EuroWordNet and Global Wordnet

Hamish Cunningham (*University of Sheffield*)
Named Entity Recognition

Ido Dagan (*Bar Ilan University*)
Learning in NLP: When can we reduce or avoid annotation cost?

Dan Cristea (*University of Iasi*)
Discourse theories and technologies

John Prager (*IBM*)
Question Answering

Inderjeet Mani (*Georgetown University*)
Automatic Summarization

RANLP 2003 was quite popular — 161 participants from 30 countries. We received 114 submissions and whilst we were delighted to have so many contributions, restrictions on the number of papers which could be presented in three days forced us to be more selective than we would have liked. However, we introduced the category of short papers allowing for discussion of ongoing research. From the papers presented at RANLP 2003 we have selected the best for this book (the acceptance ratio for the RANLP 2003 volume was about 27%), in the hope that they reflect the most significant and promising trends (and successful results) in NLP.

Keynote speakers who gave invited talks were: Shalom Lappin (King's College London), Yorick Wilks (University of Sheffield), Stephen G. Pulman (Oxford University), Inderjeet Mani (Georgetown University) and Branimir K. Boguraev (IBM T.J. Watson Research Center). The book is organised thematically. In order to allow for easier access, we have grouped the contributions according to the following topics: I. Invited talks; II. Lexical semantics and lexical knowledge acquisition; III. Tagging, parsing and syntax; IV. Information extraction; V. Text summarisation and document processing; and VI. Other NLP topics. Clearly, some papers lie at the intersection of various areas. To help the reader find his/her way we have added an index which contains major NLP

terms used throughout the volume. We have also included a list and emails of all contributors.

We would like to thank all members of the Program Committee and the additional reviewers who helped in the final stage of the review process. Without them the conference, although well organised, would not have had an impact on the development of NLP. Together they have ensured that the best papers were included in the final proceedings and have provided invaluable comments for the authors, so that the papers are 'state of the art'. The following is a list of those who participated in the selection process and to whom a public acknowledgement is due:

Le Ha An	(University of Wolverhampton)
Rie Ando	(IBM T. J. Watson Research Center)
Elisabeth Andre	(University of Augsburg)
Galia Angelova	(LMD, CLPP, Bulgarian Academy of Sciences, Sofia)
Victoria Arranz	(Universitat Politècnica de Catalunya)
Amit Bagga	(Avaya Labs Research)
Cătălina Barbu	(University of Brighton)
Lamia Belguith	(LARIS-FSEG, University of Sfax)
Branimir Boguraev	(IBM T. J. Watson Research Center)
Kalina Bontcheva	(University of Sheffield)
Svetla Boytcheva	(Sofia University)
António Branco	(University of Lisbon)
Sabine Buchholz	(Toshiba Research Europe Ltd.)
Sylviane Cardey	(University of Franche-Comté, Besançon)
Nicoletta Calzolari	(University of Pisa)
Eugene Charniak	(Brown University, Providence)
Grace Chung	(Corporation for National Research Initiatives)
Borja Navarro Colorado	(University of Alicante)
Dan Cristea	(University of Iasi)
Hamish Cunningham	(University of Sheffield)
Walter Daelemans	(University of Antwerp)
Ido Dagan	(Bar Ilan University & FocusEngine, Tel Aviv)
Robert Dale	(Macquarie University)
Hercules Dalianis	(Royal Institute of Technology, Stockholm)
Pernilla Danielsson	(University of Birmingham)
Richard Evans	(University of Wolverhampton)
Kerstin Fischer	(University of Hamburg)
Alexander Gelbukh	(National Polytechnic University, Mexico)
Ralph Grishman	(New York University)
Walther von Hahn	(University of Hamburg)
Jan Hajic	(Charles University, Prague)
Sanda Harabagiu	(University of Texas, Dallas)
Laura Hasler	(University of Wolverhampton)
Graeme Hirst	(University of Toronto)

Harold Somers	(UMIST, Manchester)
Richard Sproat	(AT&T Labs Research)
Mark Stevenson	(University of Sheffield)
Keh-Yih Su	(Behavior Design Corporation, Taiwan)
Jana Sukkarieh	(Oxford University)
Valentin Tablan	(University of Sheffield)
Hristo Tanev	(IRST, Trento)
Doina Tatar	(Babes-Bolyai University)
Kristina Toutanova	(Stanford University)
Isabel Trancoso	(INESC, Lisbon)
Jun'ichi Tsujii	(University of Tokyo)
Hans Uszkoreit	(University of Saarland)
Isabel Verdaguer	(University of Barcelona)
Karin Verspoor	(Los Alamos)
José Luis Vicedo	(University of Alicante)
Piek Vossen	(Irion Technologies BV)
Yorick Wilks	(University of Sheffield)
Zhu Zhang	(University of Michigan)
Michael Zock	(LIMSI, CNRS)

Special thanks go to Galia Angelova (Bulgarian Academy of Sciences), Nikolai Nikolov (Association for Computational Linguistics – Bulgaria), Albena Strupchanska, Milena Yankova and Ognian Kalaydjiev (Bulgarian Academy of Sciences) for the efficient local organisation. RANLP 2003 was partially supported by the European Commission with an IST FP5 Conference grant.

We believe that this book will be of interest to researchers, lecturers, graduate students and senior undergraduate students interested in Natural Language Processing and, more specifically, to those who work in computational linguistics, corpus linguistics, human language technology, translation studies, cognitive science, psycholinguistics, artificial intelligence, informatics.

We would like to acknowledge the unstinting help received from our series editor, E.F.K. Koerner, and from Ms Anke de Looper, acquision editor at John Benjamins in Amsterdam. Both of them have continued to be ever so professional.

Nicolas Nicolov produced the typesetting code for the book, utilising the TEX system with the LATEX 2$_\varepsilon$ package.

September 2004

Nicolas Nicolov
Kalina Bontcheva
Galia Angelova
Ruslan Mitkov

A Type-Theoretic Approach to Anaphora and Ellipsis Resolution

CHRIS FOX* & SHALOM LAPPIN**

*University of Essex
**King's College, London

Abstract

We present an approach to anaphora and ellipsis resolution in which pronouns and elided structures are interpreted by the dynamic identification in discourse of type constraints on their semantic representations. The content of these conditions is recovered in context from an antecedent expression. The constraints define separation types (sub-types) in Property Theory with Curry Typing (PTCT), an expressive first-order logic with Curry typing that we have proposed as a formal framework for natural language semantics.[1]

1 Introduction

We present a type-theoretic account of pronominal anaphora and ellipsis resolution within the framework of Property Theory with Curry Typing (PTCT), a first-order logic with comparatively rich expressive power achieved through the addition of Curry typing. PTCT is a fine-grained intensional logic that permits distinct propositions (and other intensional entities) to be provably equivalent. It supports functional, separation (sub), and comprehension types. It also allows a weak form of polymorphism, which seems adequate to capture type-general expressions in natural language. The proof and model theories of PTCT are classically Boolean with respect to negation, disjunction, and quantification. Quantification over functional and type variables is restricted to remain within the domain of a first-order system.

We take the resolution of pronominal anaphora to be a dynamic process of identifying the value of a type parameter with an appropriate part of the representation of an antecedent. The parameter corresponds to the specification of a sub-type condition on a quantified individual variable in PTCT, where the content of this condition is recovered from part of the antecedent expression. We propose a unified treatment for pronouns that accommodates bound variable, E-type, and donkey anaphora.

We represent ellipsis as the identification of a value of a separation type parameter for expressions in the ellipsis site. The separation type provides a pred-

[1] The research of the second author has been supported by grant number AN/2687/APN from the Arts and Humanities Research Board of the UK, and grant number RES–000–23–0065 from the Economic and Social Research Council of the UK.

icate for the bare element(s) in the ellipsis structure. The value of the parameter is constructed by abstraction on an antecedent expression. When variables corresponding to pronouns are contained in the separation type retrieved from the antecedent, typing conditions on these variables are imported into the interpretation of the elided term. Different specifications of these conditions may be possible, where each specification produces a distinct reading. Using alternative resolutions of typing constraints on the pronoun variables in the ellipsis site permits us to explain the distinction between strict vs. sloppy readings of pronouns under ellipsis. It also allows us to handle antecedent contained ellipsis without invoking syntactic mechanisms like quantifier phrase movement or semantic operations like storage. Our treatment of ellipsis is similar to the higher-order unification (HOU) analysis proposed by Dalrymple et al. (1991) and Shieber et al. (1996). However, there are a number of important differences between the two approaches which we will take up in Section 7.

In Section 2 we give a brief summary of the main features of PTCT, particularly the type definitions. More detailed descriptions are provided by Fox et al. (2002a), Fox et al. (2002b), Fox & Lappin (2003), Fox & Lappin (2004). In Section 3 we add an intensional number theory, and in Section 4 we characterize generalized quantifiers (GQs) corresponding to quantified NPs within PTCT. Section 5 gives our treatment of pronominal anaphora, and we present our account of ellipsis in Section 6. In Section 7 we compare our account of pronominal anaphora to Ranta's (1994) analysis of donkey anaphora within Martin-Löf Type Theory (MLTT) and our treatment of ellipsis to the HOU approach. Finally, in Section 8 we present some conclusions and consider directions for future work.

2 PTCT

The core language of PTCT consists of the following sub-languages:

(1) Terms $t ::= x \mid c \mid l \mid T \mid \lambda x(t) \mid (t)t$
logical constants $l ::= \hat{\wedge} \mid \hat{\vee} \mid \hat{\rightarrow} \mid \hat{\leftrightarrow} \mid \hat{\bot} \mid \hat{\forall} \mid \hat{\exists} \mid \hat{=}_T \mid \hat{\cong}_T \mid \epsilon$
(2) Types $T ::= B \mid \mathsf{Prop} \mid T \Longrightarrow S$
(3) Wff $\varphi ::= \alpha \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\varphi \rightarrow \psi) \mid (\varphi \leftrightarrow \psi) \mid (\forall x\varphi) \mid (\exists x\varphi)$
atomic wff $\alpha ::= (t =_T s) \mid \bot \mid t \in T \mid t \cong_T s \mid {}^{\mathsf{true}}t$

The language of terms is the untyped λ-calculus, enriched with logical constants. It is used to *represent* the interpretations of natural language expressions. It has no internal logic. With an appropriate proof theory, the simple language of types together with the language of terms can be combined to produce a Curry-typed λ-calculus. The first-order language of wffs is used to formulate type judgements for terms, and truth conditions for those terms judged to be in Prop.[2]

[2] Negation is defined by $\sim p =_{\mathsf{def}} p \rightarrow \bot$. Although we could formulate a constructive theory, in the following we assume rules that yield a classical Boolean version of the theory.

It is important to distinguish between the notion of a proposition itself (in the language of wff), and that of a term that *represents* a proposition (in the language of terms). $^{\mathsf{true}}(t)$ will be a true wff whenever the proposition represented by the term t is true, and a false wff whenever the proposition represented by t is false. The representation of a proposition t ($\in \mathsf{Prop}$) is distinct from its truth conditions ($^{\mathsf{true}}(t)$).

We construct a tableau proof theory for PTCT.[3] Its rules can be broken down into the following kinds.

- The basic connectives of the wff: These have the standard classical first-order behaviour.
- Identity of terms ($=$): These are the usual rules of the untyped λ-calculus with α, β and η reduction.
- Typing of λ-terms: These are essentially the rules of the Curry-typed calculus, augmented with rules governing those terms that represent propositions (Prop).
- Truth conditions for Propositions: Additional rules for the language of wffs that govern the truth conditions of terms in Prop (which represent propositions).
- Equivalence ($\cong_T$): The theory has an internal notion of extensional equivalence which is given the expected behaviour.

There are two equality notions in PTCT. $t \cong_T s$ states that the terms t, s are extensionally equivalent in type T. Extensional equivalence is represented in the language of terms by $t \mathrel{\hat{\cong}}_T s$. $t =_T s$ states that two terms are intensionally identical. The rules for intensional identity are essentially those of the $\lambda\alpha\beta\eta$-calculus. It is represented in the language of terms by $t \mathrel{\hat{=}}_T s$. It is necessary to type the intensional identity predicate in order to avoid paradoxes when we introduce comprehension types.

The rules governing equivalence and identity are such that we are able to derive $t =_T s \rightarrow t \cong_T s$ for all types inhabited by t (s), but not $t \cong_T s \rightarrow t =_T s$. As a result, PTCT can sustain fine-grained intensional distinctions among provably equivalent propositions. Therefore, we avoid the reduction of logically equivalent expressions to the same intension, a reduction which holds in classical intensional semantics, without invoking impossible worlds. Moreover, we do so within a first-order system that uses a flexible Curry typing system rather than a higher-order logic with Church typing (as in Fox et al.'s (2002c) modification of Church's (1940) Simple Theory of Types).

One possible extension that we could consider is to add a universal type Δ to the types, and rules corresponding to the following axiom.

[3] For an introduction to tableau proof procedures for first-order logic with identity see (Jeffrey 1982). Fitting (1996) presents an implemented tableau theorem prover for first-order logic with identity, and he discusses its complexity properties.

(4) UT: $x \in \Delta \leftrightarrow x = x$

Unfortunately this is inconsistent in PTCT if Prop is a type. Consider rr, where $r = \lambda x.\hat{\exists} y \in (\Delta \Longrightarrow \mathsf{Prop})[x \mathrel{\hat{=}} y \mathrel{\hat{\wedge}} \mathrel{\hat{\sim}} xy]$. However, there are other consistent extensions that can be adopted.

2.1 *Separation types*

We add $\{x \in T : \varphi'\}$ to the types, and a tableau rule that implements the following axiom.

(5) SP: $z \in \{x \in T : \varphi'\} \leftrightarrow (z \in T \wedge \varphi'[z/x])$

Note that there is an issue here concerning the nature of φ. To ensure the theory is first-order, this type needs to be term representable, so φ' must be term representable. To this end, we can define a term representable fragment of the language of wffs. First, we introduce syntactic sugar for typed quantification in the wffs.

(6) (a) $\forall_T x\varphi =_{\mathsf{def}} \forall x(x \in T \rightarrow \varphi)$
(b) $\exists_T x\varphi =_{\mathsf{def}} \exists x(x \in T \wedge \varphi)$

Wff's with these typed quantifiers, and no free-floating type judgements will then have direct intensional analogues—that is, term representations—which will always be propositions. We can define representable wffs by φ':

(7) $\varphi' ::= \alpha' \mid (\varphi' \wedge \psi') \mid (\varphi' \vee \psi') \mid (\varphi' \rightarrow \psi') \mid$
$(\varphi' \leftrightarrow \psi') \mid (\forall_T x\varphi') \mid (\exists_T x\varphi) \mid {}^{\mathsf{true}}t$
atomic representable wffs
$\alpha' ::= (t =_T s) \mid \bot \mid t \cong_T s$

The term representations of representable wffs $\lceil \alpha' \rceil$ are given as:

(8) (a) $\lceil a \wedge b \rceil = \lceil a \rceil \mathrel{\hat{\wedge}} \lceil b \rceil$
(b) $\lceil a \vee b \rceil = \lceil a \rceil \mathrel{\hat{\vee}} \lceil b \rceil$
(c) $\lceil a \rightarrow b \rceil = \lceil a \rceil \mathrel{\hat{\rightarrow}} \lceil b \rceil$
(d) $\lceil a \leftrightarrow b \rceil = \lceil a \rceil \mathrel{\hat{\leftrightarrow}} \lceil b \rceil$
(e) $\lceil a \cong_T b \rceil = \lceil a \rceil \mathrel{\hat{\cong}}_T \lceil b \rceil$
(f) $\lceil a =_T b \rceil = \lceil a \rceil \mathrel{\hat{=}}_T \lceil b \rceil$
(g) $\lceil \bot \rceil = \hat{\bot}$
(h) $\lceil {}^{\mathsf{true}}t \rceil = t$
(i) $\lceil \forall_T x.a \rceil = \hat{\forall} x \epsilon T \lceil a \rceil$
(j) $\lceil \exists_T x.a \rceil = \hat{\exists} x \epsilon T \lceil a \rceil$

Now we can express separation types as $\{x \in S.\varphi'\}$, which can be taken to be sugar for $\{x \epsilon S.\lceil \varphi' \rceil\}$. The following theorem is an immediate consequence of the recursive definition of representable wffs and their term representations.

Theorem 1 (Representability)

$\ulcorner\varphi'\urcorner \in \mathsf{Prop}$ *for all representable wffs* φ'*, and furthermore* $^{\mathsf{true}}\ulcorner\varphi'\urcorner \leftrightarrow \varphi'$.

2.2 *Comprehension types*

Usually comprehension can be derived from SP and UT. We are forgoing UT to avoid paradoxes, so we have to define comprehension independently. The same arguments apply as for SP concerning representability. We add the type $\{x : \varphi'\}$ and a tableau rule corresponding to the following axiom.

(9) COMP: $z \in \{x : \varphi\} \leftrightarrow \varphi[z/x]$

Given that COMP = SP + UT, where UT is the Universal Type $\Delta = \{x : x \hat{=} x\}$, we would derive a paradox if $=$ was not typed. This is because in PTCT Prop is a type. So rr, where $r = \lambda x.\hat{\exists} y \in (\Delta \Longrightarrow \mathsf{Prop})[x \hat{=} y \hat{\wedge} \hat{\sim} xy]$ produces a paradoxical propositional. Our use of a typed intensional identity predicate filters out the paradox because it must be possible to prove that the two expressions for which $=_T$ is asserted are of type T independently of the identity assertion. $s =_T t$ iff $s, t \in T$ and $s = t$.

2.3 *Polymorphic types*

We enrich the language of types to include type variables X, and the wffs to include quantification over types $\forall X\varphi, \exists X\varphi$. We add $\Pi X.T$ to the language of types, governed by the tableau rule corresponding to the following axiom:

(10) PM: $f \in \Pi X.T \leftrightarrow \forall X(f \in T)$

Polymorphic types permit us to accommodate the fact that natural language expressions such as coordination and certain verbs can apply as functions to arguments of different types. Note that PM is impredicative (the type quantification ranges over the types that are being defined). To avoid this, we add a language of Kinds (K) to PTCT.

(11) Kinds: $K ::= T \mid \Pi X.K$

(12) PM′: $f \in \Pi X.K \leftrightarrow \forall X(f \in K)$ where X ranges only over types.

This constraint limits quantification over types to type variables that take non-Kind types as values. Therefore, we rule out iterated type-polymorphism in which functional polymorphic types apply to polymorphic arguments. In fact, this weak version of polymporphism seems to be adequate to express the instances of multiple type assignment that occur in natural languages (van Benthem 1991).

2.4 *Final syntax*

Adopting the extensions discussed above, which do not allow the derivation of a paradox, leads to the following language.

(13) Terms $t ::= x \mid c \mid l \mid T \mid \lambda x(t) \mid (t)t$
(logical constants) $l ::= \hat{\wedge} \mid \hat{\vee} \mid \hat{\rightarrow} \mid \hat{\leftrightarrow} \mid \hat{\bot} \mid \hat{\forall} \mid \hat{\exists} \mid \hat{=}_T \mid \hat{\cong}_T \mid \epsilon$
(14) Types $T ::= B \mid \mathsf{Prop} \mid T \Longrightarrow S \mid X \mid \{x \in T.\varphi'\} \mid \{x.\varphi'\}$
(15) Kinds $K ::= T \mid \Pi X.T$
(16) Wff $\varphi ::= \alpha \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\varphi \rightarrow \psi) \mid (\varphi \leftrightarrow \psi)$
$\mid (\forall x\varphi) \mid (\exists x\varphi) \mid (\forall X\varphi) \mid (\exists X\varphi)$
(atomic wff) $\alpha ::= (t =_T s) \mid \bot \mid t \in K \mid t \cong_T \mid {}^{\mathsf{true}}t$

where φ' is as defined in section 2.1.

2.5 *A model theory for PTCT*

In order to give a model for PTCT, we first need a model of the untyped λ-calculus. This will form the model for PTCT's language of terms. Here we present Meyer's model (Meyer 1982).

Definition 1 (General Functional Models) *A functional model is a structure of the form $\mathcal{D} = \langle D, [D \rightarrow D], \Phi, \Psi \rangle$ where*

- *D is a non-empty set,*
- *$[D \rightarrow D]$ is some class of functions from D to D,*
- *$\Phi : D \rightarrow [D \rightarrow D]$,*
- *$\Psi : [D \rightarrow D] \rightarrow D$,*
- *$\Psi(\Phi(d)) = d$ for all $d \in D$*

We can interpret the calculus as follows (g is an assignment function from variables to elements of D):

(17) $[\![x]\!]_g = g(x)$
$[\![\lambda x.t]\!]_g = \Psi(\lambda d.[\![t]\!]_{g[d/x]})$
$[\![ts]\!]_g = \Phi([\![t]\!]_g)[\![s]\!]_g$

This interpretation exploits the fact that Φ maps every element of D into a corresponding function from D to D, and Ψ maps functions from D to D into elements of D. Note we require that functions of the form $\lambda d.[\![t]\!]_{g[d/x]}$ are in the class $[D \rightarrow D]$ to ensure that the interpretation is well defined. In the case where we permit constant terms, then we can add the clause (i assigns elements from D to constants):

(18) $[\![c]\!]_g = i(c)$

Theorem 2 *If $t = s$ in the extensional untyped λ-calculus (with ξ and η), then $[\![t]\!]_g = [\![s]\!]_g$ for each assignment g.*

Proof: By induction on the derivations. □

A model $\mathcal{M}$ of PTCT is constructed on the basis of a simple extensional model of the untyped λ-calculus (Meyer 1982, Barendregt 1984, Turner 1997), with additional structure added to capture the type rules and the relation between the sublanguages of PTCT. On the basis of the full proof and model theories, we prove the soundness and completeness of PTCT.[4]

3 An intensional number theory

We add an intensional number theory to PTCT, incorporating rules corresponding to the axioms in (23).

(19) Terms: $0 \mid succ \mid pred \mid add \mid mult \mid \hat{most} \mid \mid \cdot \mid_B$
(20) Types: Num
(21) Wffs: $zero(t) \mid t \cong_{\mathsf{Num}} t' \mid t <_{\mathsf{Num}} t' \mid most(p)(q)$
(22) Axioms for Num: The usual Peanoaxioms, adapted to PTCT
(23) Axioms for $<_{\mathsf{Num}}$:
  (a) $y \in \mathsf{Num} \rightarrow 0 <_{\mathsf{Num}} succ(y)$
  (b) $x \in \mathsf{Num} \rightarrow x \not<_{\mathsf{Num}} 0$
  (c) $x \in \mathsf{Num} \wedge y \in \mathsf{Num} \rightarrow (succ(x) <_{\mathsf{Num}} succ(y) \leftrightarrow x <_{\mathsf{Num}} y)$

The model theory can be extended in a straightforward way to support these new rules of the proof theory.

When we incorporate the intensional number theory into PTCT we lose completeness of the proof theory. However, it is important to recognize that incompleteness sets in only when tableau rules that encode number-theoretic inferences are applied. The basic logic and type system of PTCT without these rules and their corresponding definitions in the model theory remains complete.

4 Representing proportional generalized quantifiers in PTCT

By defining the cardinality of properties, we can express the truth conditions of proportional quantifiers in PTCT. The cardinality of properties $|p|_B$ is formulated as follows.

(24) $p \in (B \Longrightarrow \mathsf{Prop}) \wedge \sim \exists x(x \in B \wedge {}^{\mathsf{true}}px) \rightarrow \quad |p|_B \cong_{\mathsf{Num}} 0$
(25) $p \in (B \Longrightarrow \mathsf{Prop}) \wedge b \in B \wedge {}^{\mathsf{true}}pb \rightarrow$
$|p|_B \cong_{\mathsf{Num}} \quad add(|\lambda x(px \mathbin{\hat{\wedge}} \sim x \mathbin{\hat{=}}_B b)|_B)(succ(0))$

The cardinality of types can be defined in a similar way. We represent $most(p)(q)$ as follows:

[4] The full proof and model theories for PTCT, and the proofs for soundness and completeness are presented by Fox & Lappin (2004).

(26) $p \in (B \Longrightarrow \mathsf{Prop}) \wedge q \in (B \Longrightarrow \mathsf{Prop}) \rightarrow$
$most(p)(q) \leftrightarrow$
$|\{x \in B.^{\mathsf{true}} px \wedge \sim {}^{\mathsf{true}} qx\}|_B <_{\mathsf{Num}} |\{x \in B.^{\mathsf{true}} px \wedge {}^{\mathsf{true}} qx\}|_B$

Given that PTCT is a first-order theory in which all quantification is limited to first-order variables, this characterization of $most$ effectively encodes a higher-order GQ within a first-order system.

5 A type-theoretical approach to anaphora

We combine our treatment of generalized quantifiers with our characterisation of separation types to provide a unified type-theoretic account of anaphora. We assume that all quantified NPs are represented as cardinality relations on the model of our treatment of *most*. Pronouns are represented as appropriately typed free variables. If the free variable is within the scope of a set forming operator that specifies a sub-type and it meets the same typing constraints as the variable bound by the operator, the variable can be interpreted as bound by the operator through substitution under α identity. This interpretation yields the bound reading of the pronoun.

(27) Every man loves his mother.
(28) $|\{x \in B.^{\mathsf{true}} man'(x) \wedge {}^{\mathsf{true}} love'(x, mother\text{-}of'(y))\}|_B$
$\cong_{\mathsf{Num}} |\{x \in B.^{\mathsf{true}} man'(x)\}|_B$
$\rightarrow$
$|\{x \in B.^{\mathsf{true}} man'(x) \wedge {}^{\mathsf{true}} love'(x, mother\text{-}of'(x))\}|_B$
$\cong_{\mathsf{Num}} |\{x \in B.^{\mathsf{true}} man'(x)\}|_B$

Representations of this kind are generated by compositional semantic operations as described by (for example) Lappin (1989), and Lappin & Francez (1994).

When the pronoun is interpreted as dependent upon an NP which does not bind it, we represent the pronoun variable as constrained by a separation type parameter whose value is supplied by context. Generally, the value of this type parameter is determined in two parts. The initial type membership condition is imported directly from the antecedent corresponding to the GQ. The wff part of the separation type corresponds to the relation between the restriction and the predication in the antecedent clause. In the default case, the pronoun variable is bound by a universal quantifier in the language of wffs.

(29) Every student arrived.
(30) $|\{x \in B.^{\mathsf{true}} student'(x) \wedge {}^{\mathsf{true}} arrived'(x)\}|_B$
$\cong_{Num} |\{x \in B.^{\mathsf{true}} student'(x)\}|_B$
(31) They sang.
(32) $\forall y \in A.({}^{\mathsf{true}} sang'(y))$
where $A = \{x \in B.^{\mathsf{true}} student'(x) \wedge {}^{\mathsf{true}} arrived'(x)\}$

In the case of proper names and existentially quantified NP antecedents we obtain the following.

(33) John arrived.
(34) $^{\mathsf{true}}arrived'(john)$
(35) He sang.
(36) $\forall y \in A.(^{\mathsf{true}}sang'(y))$
where $A = \{x \in B.^{\mathsf{true}}x \hat{=}_B john \wedge {}^{\mathsf{true}}arrived'(x)\}$
(37) Some man arrived.
(38) $|\{x \in B.^{\mathsf{true}}man'(x) \wedge {}^{\mathsf{true}}arrived'(x) \wedge {}^{\mathsf{true}}\phi(x)\}|_B >_{Num} 0$
(39) He sang.
(40) $\forall y \in A.(^{\mathsf{true}}sang'(y))$
where $A = \{x \in B.^{\mathsf{true}}man'(x) \wedge {}^{\mathsf{true}}arrived'(x) \wedge {}^{\mathsf{true}}\phi(x)\}$

ϕ is a predicate that is specified in context and uniquely identifies a man who arrived in that context.

We handle donkey anaphora in PTCT through a type constraint on the variable corresponding to the pronoun.

(41) Every man who owns a donkey beats it.
(42) $|\{x \in B.^{\mathsf{true}}man'(x) \wedge$
$(|\{y \in B.^{\mathsf{true}}own'(x,y) \wedge {}^{\mathsf{true}}donkey'(y)\}|_B >_{Num} 0)$
$\wedge \forall z \in A(^{\mathsf{true}}beat'(x,z))\}|_B$

$\cong_{Num}$
$|\{x \in B.^{\mathsf{true}}man'(x) \wedge (|\{y \in B.^{\mathsf{true}}own'(x,y) \wedge {}^{\mathsf{true}}donkey'(y)\}|_B$
$>_{Num} 0)\}|_B$
where
$A = \{y \in B.^{\mathsf{true}}own'(x,y) \wedge {}^{\mathsf{true}}donkey'(y)\}$

The representation asserts that every man who owns at least one donkey beats all of the donkeys that he owns.

Our type-theoretic account of donkey anaphora is similar in spirit to the E-type analysis proposed by Lappin & Francez (1994). There is, however, an important difference. Lappin & Francez (1994) interpret an E-type pronoun as a choice function from the elements of an intersective set specified by the clause containing the antecedent NP to a range of values determined by this NP. Both the domain and range of the function are described informally in terms of the semantic representation of the antecedent clause. On the type-theoretic approach proposed here the interpretation of the E-type pronoun is specified explicitly through type constraints on variables in the semantic representation language. Therefore our account provides a more precise and properly formalized treatment of pronominal anaphora.

We can generate existential readings of donkey sentences (Pelletier & Schubert 1989) by treating the principle that the free variable representing a pronoun

is bound by a universal quantifier as defeasible. We can then substitute an existential for the universal quantifier.

(43) Every person who had a quarter put it in a parking meter.

(44) $|\{x \in B.^{\mathsf{true}} person'(x) \wedge$
$(|\{y \in B.^{\mathsf{true}} had'(x,y) \wedge {}^{\mathsf{true}} quarter'(y)\}|_B >_{Num} 0)$
$\wedge \exists z \in A({}^{\mathsf{true}} put\text{-}in\text{-}meter'(x,z))\}|_B$
$\cong_{Num}$
$|\{x \in B.^{\mathsf{true}} person'(x) \wedge$
$(|\{y \in B.^{\mathsf{true}} had'(x,y) \wedge {}^{\mathsf{true}} quarter'(y)\}|_B$
$>_{Num} 0)\}|_B$
where $A = \{y \in B.^{\mathsf{true}} had'(x,y) \wedge {}^{\mathsf{true}} quarter'(y)\}$

This representation asserts that every person who had a quarter put at least one quarter that he/she had in a parking meter.

The default presence of a universal quantifier in the PTCT representation of a donkey pronoun is an instance of a pragmatic maximality condition of the kind that Lappin & Francez (1994) invoke to explain the preferred readings of sentences like (41). As they observe, lexical semantic and pragmatic factors can override a maximality constraint in cases like (43). We represent the suspension of the maximality requirement by substituting an existential for the universal quantifier binding the variable corresponding to the pronoun in these sentences.

6 Ellipsis

Let S be a parameter that is instantiated by separation types. We can represent a clause containing an elided VP like (45) as (46).

(45) John sings, and Mary does too.

(46) ${}^{\mathsf{true}} sings'(john) \wedge mary \in S$

Assuming that $mary$ and $john$ are both of type B, we can abstract on $john$ to obtain the separation type $\{x \in B.^{\mathsf{true}} sings'(x)\}$ from the antecedent in order to resolve S. This yields the desired interpretation of the elided clause in (47).[5]

(47) ${}^{\mathsf{true}} sings'(john) \wedge {}^{\mathsf{true}} sings'(mary)$

We have not introduced product types into PTCT, but we are assuming that all predicate types are curried functions. For simplicity of notation we represent transitive and ditransitive verbs as multi-argument functional expressions, but we continue to assume that they are interpreted as a sequence of curried

[5] The presentation adopted here requires a slight change to PTCT as it is formulated elsewhere (Fox et al. 2002a, Fox et al. 2002b, Fox & Lappin 2003) in order to allow free-floating type judgements within the language of terms. This extension is not problematic provided appropriate restrictions are observed (Fox & Lappin, to appear).

functions. Similarly we assume that separation types corresponding to curried function predicate can be built up through curried function application.

Our treatment of VP ellipsis extends directly to gapping (48).[6]

(48) Mary reviewed Principia and Max Ulysses.
(49) $^{\mathsf{true}}reviewed'(mary, principia) \wedge (max, ulysses) \in S$
(50) $S = \{(x, y) \in B \otimes B.^{\mathsf{true}}reviewed'(x, y)\}$
(51) $^{\mathsf{true}}reviewed'(john, principia) \wedge {}^{\mathsf{true}}reviewed'(max, ulysses)$

It also applies to pseudogapping (52).[7]

(52) Max introduced Rosa to Sam before
Bill did Mary to John.
(53) $^{\mathsf{true}}introduced'(max, rosa, sam)$
before
$(bill', mary, john) \in S$
(54) $S = \{(x, y, z) \in B \otimes B \otimes B.$
$^{\mathsf{true}}introduced'(x, y, z)\}$
(55) $^{\mathsf{true}}introduced'(max, rosa, sam)$
before
$^{\mathsf{true}}introduced'(bill, mary, john)$

If we combine our treatment of ellipsis with our account of pronominal anaphora, the representation of the distinction between strict and sloppy readings of pronouns under ellipsis is straightforward.

(56) John loves his mother, and Bill does too.
(57) $\forall x \in A(^{\mathsf{true}}loves'(john, mother\text{-}of'(x)) \wedge bill \in S$

Let S be defined as follows.

$$S = \{y \in B.\forall x \in A(^{\mathsf{true}}loves'(y, mother\text{-}of'(x))\}$$

If the type parameter A on the variable x is specified as $\{w \in B.^{\mathsf{true}}w \hat{=}_B john\}$ in the antecedent clause prior to the resolution of S, then a strict reading of the pronoun results. If A is determined after the value of S is identified, then it can be taken as $\{w \in B.^{\mathsf{true}}w \hat{=}_B bill\}$, which provides the sloppy reading.

Finally, consider the antecedent contained ellipsis (ACE) structure:

(58) Mary read every book that John did.

Restrictive relative clauses modify head nouns of NPs. Therefore, it is reasonable to impose the condition that the conjunct corresponding to a restrictive relative clause in the propositional part of the sub-type expression of a GQ contain an

[6] Here we use product types. Product types can be added to PTCT (Fox & Lappin, to appear), or the examples can be represented using an equivalent curried form.

[7] Here we assume there is some suitable treatment of the temporal ordering of the circumstances described by the propositions p and q in the expression p before q without further elaboration.

occurrence of the variable bound by the set operator of the sub-type. This is, in effect, a non-vacuousness constraint on relative clause modification. It requires that a relative clause be interpreted as a modifier that contributes a restriction to the head noun. Given this constraint the representation of (58) is:

(59) $|\{x \in B.^{\mathsf{true}} book'(x) \wedge (john, x) \in S) \wedge {}^{\mathsf{true}} read'(mary, x)\}|_B$
$\cong_{\mathsf{Num}} |\{x \in B.^{\mathsf{true}} book'(x) \wedge (john, x) \in S\}|_B$

Taking the conjunct of (59) that corresponds to the matrix clause as the antecedent and abstracting over both its arguments we obtain the separation type specified in (60). This yields (61) as the interpretation of (58).[8]

(60) $S = \{(y, w).^{\mathsf{true}} read'(y, w)\}$

(61) $|\{x \in B.^{\mathsf{true}} book'(x) \wedge {}^{\mathsf{true}} read'(john, x) \wedge {}^{\mathsf{true}} read'(mary, x)\}|_B$
$\cong_{\mathsf{Num}} |\{x \in B.^{\mathsf{true}} book'(x) \wedge {}^{\mathsf{true}} read'(john, x)\}|_B$

Statement (61) asserts that every book that John read Mary read, which is the intended reading of (58).

We have generated this interpretation without using a syntactic operation of quantifier raising, as in the analysis of Fiengo & May (1994) or a semantic procedure of storage, as in the HOU treatment of Dalrymple et al. (1991). We also do not require a syntactic trace (Lappin 1996) or a SLASH feature (Lappin 1999) in the ellipsis site. The presence of the variable bound by the set operator of the sub-type as the second argument of the function which assigns a value to the elided PTCT expression is motivated by a general condition on the representation of restrictive relative clauses as non-vacuous conjuncts in a GQ.

7 Comparison with other type-theoretical approaches

Ranta develops an analysis of anaphora within Martin-Löf Type Theory (MLTT) (Ranta 1994). He represents donkey sentences as universal quantification over product types.[9]

(62) $\Pi z : ((\Sigma x : man)(\Sigma y : donkey)(x\ owns\ y))$
$(p(z)\ beats\ p(q(z))$

In this example, z is a variable over product pairs, and p and q are left and right projections, respectively, on the product pair, where

(63) (a) $p(z) : man$
(b) $q(z) : (\Sigma y : donkey)(p(z)\ owns\ y)$
(c) $p(q(z)) : donkey$

[8] For simplicity we suppress typing on the variables y and w here.

[9] Note that here expressions of the form $\Pi x : (T)(S)$ denote a dependent product type. This is not to be confused with PTCT's polymorphic types, whose form ($\Pi X.T$) is superficially similar.

(d) $q(q(z)) : (p(z)\ owns\ p(q(z)))$.

Ranta's account does not generate the existential reading of donkey sentences (Pelletier & Schubert 1989).

As Ranta acknowledges, his universal quantification-over-pairs analysis follows Discourse Representation Theory (DRT) (Kamp & Reyle 1993) in inheriting the proportionality problem in a sentence like the following (Heim 1990, Kadmon 1990).

(64) Most men who own a donkey beat it.

On the universal quantification-over-pairs account of donkey anaphora, contrary to the desired interpretation, the sentence is true in a model in which ten men own donkeys, nine men own a single donkey each and do not beat it, while the tenth man owns ten donkeys and beats them all. On the preferred reading the sentence is false in this model.

Ranta cites Sundholm's (1989) solution to the proportionality problem. This suggestion involves positing a second quantifier *most* on product pairs $(\Sigma x : A)(B(x))$ that is interpreted as applying an A injection to the pairs, where this injection identifies only the first element of each pair as within the domain of quantification. Defining an additional mode of quantification as a distinct reading of *most* in order to generate the correct interpretation of (64) would seem to be an ad hoc approach to the difficulty. There is no evidence for taking *most* as ambiguous between two quantificational readings beyond the need to avoid the inability of the quantification-over-pairs analysis to yield the correct results for this case. Assuming two modes of quantification adds considerable complication to the type-theoretic approach to anaphora without independent motivation.

The proportion problem does not arise on our account. *Most* is represented as a cardinality relation (GQ) in which quantification is over the elements of the set corresponding to the subject restriction rather than over pairs. Therefore the sentence is evaluated as false in the model that creates difficulties for DRT and for Ranta, without the need to adopt additional devices.

On the HOU approach to ellipsis proposed by Dalrymple et al. (1991) and Shieber et al. (1996), the elided predicate is represented as a higher-order variable, which is unified with a lambda term obtained from the antecedent clause through abstraction over the arguments that correspond to the bare arguments of the ellipsis site. This term is then applied to the bare arguments to produce an interpretation of the elided structure.

Our PTCT-based analysis of ellipsis is similar in approach to the HOU view. In both cases correspondences are set up between a sequence of phrases in an ellipsis site and an antecedent clause, and a predicate term is abstracted from the antecedent for application to elements in the elided clause. However, while HOU solves an equation with a higher-order variable to obtain a lambda expression, our account uses a parameter that is resolved to a separation type expression.

In our model theory, type variables take terms as values. Therefore, even if a separation type parameter is construed as a variable of PTCT, we remain within the first-order resources of PTCT.

In addition, HOU requires storage to extract a quantified NP from its antecedent-contained position in the semantic representation of an ACE structure. By contrast, we are able to interpret these NPs *in situ* by virtue of the presence of a bound variable in the part of a sub-type in a GQ representation that corresponds to the relative clause of the ACE.

8 Conclusions and future work

We have developed type-theoretic treatments of pronominal anaphora and ellipsis within the framework of PTCT, a first-order fine-grained intensional logic with flexible Curry typing. Our account of anaphora has wider empirical coverage than Ranta's (1994) MLTT analysis. Our account of ellipsis avoids the higher-order variables of HOU, and we do not require an operation of storage to handle ACE structures.

The application of PTCT to anaphora and ellipsis illustrates its considerable expressive resources. The primary advantage of PTCT is the fact that it provides the expressiveness of a higher-order system with rich typing while remaining a first-order logic with limited formal power.

We have provided a model theory for PTCT using extensional models for the untyped λ-calculus enriched with interpretations of Curry types. The restrictions that we impose on comprehension types, quantification over types, and the relation between the three sublanguages of PTCT ensure that it remains a first-order system in which its enriched expressive power comes largely through quantification over terms and the representation of types as terms within the language.

In future work we will be investigating the possibility of incorporating product types into PTCT without taking it out of the class of first-order systems and also determine the most appropriate way of incorporating the representation of free-floating type judgements in the language of terms. Product types will permit us to simplify our representations of k-ary predicates and the sub-types defined in terms of them. They will also permit us to capture dependent types, which we require to deal with certain kinds of anaphora, such as donkey pronouns in conditional sentences.

We will consider a property-theoretic variant of the treatment of anaphora and ellipsis which exploits properties and application rather than the types and type-membership used in the type-theoretic treatment presented in this paper. We will then examine extensions that establish correspondences between types and properties. One aim of this would be to show an equivalence between the type-theoretic and property-theoretic approaches to anaphora and ellipsis.

We will explore PTCT as a semantic representation language in implemented systems of natural language interpretation. As part of this research we will be constructing a theorem prover that uses PTCT's tableau proof theory. We will also investigate the implementation of our proposed approaches to anaphora and ellipsis resolution.

REFERENCES

Barendregt, Henk. 1984. *The Lambda Calculus: Its Syntax and Semantics* (vol. 103, Second edition). Amsterdam: North Holland.

Church, Alonzo. 1940. "A Formulation of the Simple Theory of Types". *Journal of Symbolic Logic* 5:56-68.

Dalrymple, Mary, Stuart Shieber & Fernando Pereira. 1991. "Ellipsis and Higher-Order Unification". *Linguistics and Philosophy* 14:399-452.

Fiengo, Robert & Robert May. 1994. *Indices and Identity*. Cambridge, Mass.: MIT Press.

Fitting, Melvin. 1996. *First-Order Logic and Automated Theorem Proving*. Berlin: Springer.

Fox, Chris, Shalom Lappin & Carl Pollard. 2002a. "First-Order Curry-Typed Logic for Natural Language Semantics". *Proceedings of the 7th International Workshop on Natural Language Understanding and Logic Programming* ed. by S. Winter, 175-192. Copenhagen: University of Copenhagen.

Fox, Chris, Shalom Lappin & Carl Pollard. 2002b. "Intensional First-Order Logic with Types". *Proceedings of the 7th Symposium for Logic and Language* ed. by G. Alberti, K. Balough & P. Dekker, 47-56. Pecs, Hungary: Univ. of Pecs.

Fox, Chris, Shalom Lappin & Carl Pollard. 2002c. "A Higher-Order Fine-grained Logic for Intensional Semantics". *Proceedings of the 7th Symposium for Logic and Language* ed. by G. Alberti, K. Balough & P. Dekker, 37-46. Pecs, Hungary: Univ. of Pecs.

Fox, Chris & Shalom Lappin. 2003. "Doing Natural Language Semantics in an Expressive First-Order Logic with Flexible Typing". *Proceedings of Formal Grammar 2003* ed. by G. P. G. Jaeger, P. Monachesi & S. Wintner, 89-102. Vienna: Technical University of Vienna.

Fox, Chris & Shalom Lappin. 2004. "An Expressive First-Order Logic with Flexible Typing for Natural Language Semantics". *Logic Journal of the Interest Group in Pure and Applied Logics (IGPL)*, vol. 12. Oxford University Press.

Fox, Chris & Shalom Lappin. To appear. *Foundations of Intensional Semantics*. Oxford: Blackwell.

Heim, Irene. 1990. "E-type Pronouns and Donkey Anaphora". *Linguistics and Philosophy* 13:137-177.

Jeffrey, Richard. 1982. *Formal Logic: Its Scope and Limits*. New York: McGraw-Hill.

Kadmon, Nirit. 1990. "Uniqueness". *Linguistics and Philosophy* 13:237-324.

Kamp, Hans & Uwe Reyle. 1993. *From Discourse to Logic*. Dordrecht: Kluwer.

Lappin, Shalom. 1989. "Donkey Pronouns Unbound". *Theoretical Linguistics* 15:263-286.

Lappin, Shalom. 1996. "The Interpretation of Ellipsis". *Handbook of Contemporary Semantic Theory* ed. by S. Lappin, 145-175. Oxford: Blackwell.

Lappin, Shalom. 1999. "An HPSG Account of Antecedent Contained Ellipsis". *Fragments: Studies in Ellipsis and Gapping* ed. by S. Lappin & E. Benmamoun, 68-97. New York: Oxford University Press.

Lappin, Shalom & N. Francez. 1994. "E-type Pronouns, i-Sums, and Donkey Anaphora". *Linguistics and Philosophy* 17:391-428.

Meyer, Albert R. 1982. "What is a Model of the Lambda Calculus?" *Information and Control* 52:87-122.

Pelletier, Francis Jeffrey & Lenhart Schubert. 1989. "Generically Speaking". *Propterties, Types, and Meaning* (vol. 2) ed. by G. Chiercihia, B. Partee & R. Turner, 141-167. Dordrecht: Kluwer.

Ranta, Aarne. 1994. *Type Theoretic Grammar*. Oxford: Oxford University Press.

Shieber, Stuart, Fernando Pereira & Mary Dalrymple. 1996. "Interactions of Scope and Ellipsis". *Linguistics and Philosophy* 19:527-552.

Sundholm, Goeran. 1989. "Constructive Generalised Quantifiers". *Synthese* 79:1-12.

Turner, Raymond. 1997. "Types". *Handbook of Logic and Language* ed. by Johan van Benthem & A. ter Meulen, 535-586. Amsterdam: North-Holland.

van Benthem, Johan. 1991. *Language in Action* (= *Studies in Logic*, 130). Amsterdam: North-Holland.

Human Dialogue Modelling Using Machine Learning

YORICK WILKS, NICK WEBB, ANDREA SETZER,
MARK HEPPLE & ROBERTA CATIZONE

University of Sheffield

Abstract

We describe two major dialogue system segments: first we discuss a Dialogue Manager which uses a representation of stereotypical dialogue patterns that we call Dialogue Action Frames and which, we believe, generate strong and novel constraints on later access to incomplete dialogue topics. Secondly, we describe an analysis module that learns to assign dialogue acts from corpora, but on the basis of limited quantities of data, and up to what seems to be some kind of limit on this task, a fact we also discuss.

1 Introduction

Computational modelling of human dialogue is an area of NLP where there are still a number of open research issues about how such modelling should best be done. Most research systems so far have been largely hand-coded, inflexible representations of dialogue states, implemented as some form of finite state or other rule-based machine. These approaches have addressed robustness issues within spoken language dialogue systems by limiting the range of the options and vocabulary available to the user at any given stage in the dialogue. They have, by common agreement, failed to capture much of the flexibility and functionality inherent in human-human communication, and the resulting systems have far less than optimal conversational capability and are neither pleasant nor natural to use. However, many of these low-functionality systems have been deployed in the market, in domains such as train reservations.

On the other hand, more flexible, conversationally plausible models of dialogue, such as those based on planning (Allen et al. 1995) are knowledge rich, and require very large amounts of manual annotation to create. They model individual communication actions, which are dynamically linked together into plans to achieve communicative goals. This method has greater scope for reacting to user input and correcting problems as they occur, but has never placed emphasis on either implementation or evaluation.

The model we wish to present occupies a position between these two approaches: full planning systems and turned-based dialogue move engines. We contend that larger structures are necessary to represent the content and context provided by mini-domains or meta-dialogue processes as opposed to modelling only turn taking. The traditional problems with our position are: how to obtain the data that such structures (which we shall call Dialogue Action Frames or

DAFs) contain, and how to switch rapidly between them in practice, so as not to be stuck in a dialogue frame inappropriate to what a user has just said. We shall explain their functioning within an overall control structure that stacks DAFs, and show that we can leave a DAF in any dialogue state and return to it later if appropriate, so that there is no loss of flexibility, and we can retain the benefits of larger scale dialogue structure. For now, DAFs are hand-coded but ultimately we are seeking to learn them from annotated dialogue corpora. In so doing, we hope to acquire those elements of human-human communication which may make a system more conversationally plausible.

A second major area that remains unsettled in dialogue modelling is the degree to which its modules can be based directly on abstractions from data (abstractions usually obtained by some form of Machine Learning) as significant parts of NLP have been over the last fifteen years. We shall describe a system for learning the assignment of dialogue acts (DAs) and semantic content directly from corpora, while noting the following difficulty: in the five years since Samuels et al. (1998) first demonstrated such a technique based on Transformation-Based Learning the figures obtained have remained obstinately in the area of 65%+ and not risen towards the Nineties as has been the case in other, perhaps less complex, areas of linguistic information processing, such as part-of-speech tagging.

In the model that follows, we hypothesise that the information content of DAs may be such that some natural limit has appeared to their resolution by the kinds of ngram-based corpus analysis used so far, and that the current impasse, if it is one, can only be solved by realising that higher level dialogue structures in the DM will be needed to refine the input DAs, that is, by using the inferential information in DAFs, along with access to the domain model. This hypothesis, if true, explains the lack of progress with a purely data-driven research in this area and offers a concrete hybrid model. This process could be seen as one of the correction or reassignment of DA tags to input utterances in a DM, where a higher level structure will be able to chose from some (possibly ordered) list of alternative DA assignments as selected by our initial process.

2 Modality independent dialogue management

The development of our Dialogue Management strategies has occured largely within the COMIC (Conversational Multimodal Interaction with Computers)[1] project whose object is to build a cooperative multi-modal dialogue system which aids the user in the complex task of designing a bathroom, and a system to be deployed in a showroom scenario. A central part of this system is the Dialogue and Action Manager (DAM).

[1] See `http://www.hcrc.ed.ac.uk/comic/`

There is as yet no consensus as to whether a DAM should be expressed simply as a finite-state automaton, a well understood and easy to implement representation, or utilise more complex, knowledge-based approaches such as the planning mechanism employed by systems such as TRAINS.

The argument between these two views, at bottom, is about how much stereotopy one expects in a dialogue and which is to say, is it how much is it worth collecting all rules relevant to a subtopic together, within some larger structure or partition? Stereotopy in dialogue is closely connected to the notion of system-initiative or top-down control, which is strongest in "form-filling" systems and weakest in chatbots. If there is little stereotopy in dialogue turn ordering, then any larger frame-like structure risks being over-repetitious, since all possibilities must be present at many nodes. If a system must always be ready to change topic in any state, it can be argued, then what is the purpose of being in a higher level structure that one may have to leave? The answer to that it is possible to be always ready to change topic but to continue on if change is not forced: As with all frame-like structures since the beginning of AI, they express no more than defaults or preferences.

The WITAS system (Lemon et al. 2001) was initially based on networks of ATN (Augmented Transition Network) structures, stacked on one of two stacks. In the DAM described below we also opt for an ATN-like system which has as its application mechanism a single stack (with one slight modification) of DAF's (Dialogue Action Frames) and suggest that the WITAS argument for abandoning an ATN-type approach (namely, that structure was lost when a net is popped) is easily overcome. We envisage DAFs of radically different sizes and types: complex ones for large scale information eliciting tasks, and small ones for dialogue control functions such as seeking to reinstate a topic.

Our argument will be that the simplicity and perspicuity of this (well understood and easily written and programmed) virtual machine (at least in its standard form) has benefits that outweigh its disadvantages, and in particular the ability to leave and return to a topic in a natural and straightforward way.

2.1 *DAFs: A proposed model for DAM*

We propose a single pop-push stack architecture that loads structures of radically differing complexities but whose overall forms are DAFs. The algorithm to operate such a stack is reasonably well understood, though we will suggest below one amendment to the classical algorithm, so as to deal with a dialogue revision problem that cannot be dealt with by structure nesting.

The general argument for such a structure is its combination of power, simplicity and perspicuity. Its key language-relevant feature (known back to the time of Woods (1997) in syntactic parsing) is the fact that structures can be pushed down to any level and re-entered via suspended execution, which allows nesting

of topics as well as features like barge-in and revision with a smooth and clear return to unfinished materials and topics. Although, in recursive syntax, incomplete parsing structures must be returned to and completed, in dialogue one could argue that not all incomplete structures should be re-entered for completion since it is unnatural to return to every suspended topic no matter how long suspended, unless, that is, the suspended structure contains information that must be elicited from the user. There will be DAFs corresponding to each of the system-driven sub-tasks which are for eliciting information and whose commands write directly to the output database. There will also be DAFs for standard Greetings and Farewells, and for complex dialogue control tasks like revisions and responses to conversational breakdowns. A higher granularity of DAFs will express simple dialogue act pairs (such as QA) which can be pushed at any time (from user initiative) and will be exhausted (and popped) after an SQL query to the COMIC database.

The stack is preloaded with a (default) ordered set of system initiative DAFs, with Greeting at the top, Farewell at the bottom and such that the dialogue ends with maximum success when these and all the intermediate information eliciting DAFs for this task have been popped. This would be the simplest case of a maximally cooperative user with no initiative whatever; he may be rare but must be catered for if he exists.

An obvious problem arises here, noted in earlier discussion, which may require that we adapt this overall DAM control structure:

If the user proposes an information eliciting task before the system does (e.g., in a bathroom world, we suppose the client wants to discuss tile-colour-choice before that DAF is reached in the stack) then that structure must immediately be pushed onto the stack and executed till popped, but obviously its copy lower in the stack must not be executed again when it reaches the top later on. The integrity of the stack algorithm needs to be violated only to the extent that any task-driven structure at the top of the stack is only executed from its initial state if the relevant part of the database is incomplete.

However, a closely related, issue (and one that caused the WITAS researchers to change their DAM structure) is the situation where a user-initiative forces the revision/reopening of a major topic already popped from the stack; e.g., in a bathroom world, the user has chosen pink tiles but later, and at her own initiative, decides she would prefer blue and initiates the topic again. This causes our proposal no problems: the tile-colour-choice DAF structure is pushed again (empty and uninstantiated) but with an entry subnetwork that can check the database, see it is complete, and begin the subdialogue in a way that responses show the system knows a revision is being requested. It seems clear to us that a simple stack architecture is proof against arguments based on the need to revisit popped structures, provided the system can distinguish this case (as user initiative) from the last (a complete structure revisited by system initiative).

A similar device will be needed when a partly executed DAF on the stack is re-entered after an interval; a situation formally analogous to a very long syntactic dependency or long range co-reference. In such cases, a user should be asked whether he wishes to continue the suspended network (to completion). It will be an experimental question later, when data has been generated, whether there are constraints on access to incomplete DAFs that will allow them to be dumped from the top of the stock unexecuted, provided they contain no unfilled requests for bathroom choices.

We expect later to build into the DAM an explicit representation of plan tasks, and this will give no problem to a DAF since recursive networks can be, and often have been, a standard representation of plans, which makes it odd that some redesigners of DAM's have argued against using ATNs as DAM models, wrongly identifying them with low-level dialogue grammars, rather than, as they are, structures (ATNs) more general than those for standard plans (RTNs).

3 Learning to annotate utterances

In the second part of this paper, we will focus on some experiments on modelling aspects of dialogue directly from data. In the joint EU-, US- funded project AMITIES[2] we are building automated service counters for telephone-based interaction, by using large amounts of recorded human-human data.

Initially, we report on some experiments on learning the analysis part of the dialogue engine; that is, that part which converts utterances to dialogue act and semantic units.

Two key annotated corpora, which have formed the basis for work on dialogue act modelling are of particular relevance here: first, the VERBMOBIL corpus , which was created within the project developing the VERBMOBIL speech-to-speech translation system, and secondly, the SWITCHBOARD corpus (Jurafsky et al. 1998). Of the two, SWITCHBOARD has generally been considered to present a more difficult problem for accurate dialogue act modelling, partly because it has been annotated using a total of 42 distinct dialogue acts, in contrast to the 18 used in the VERBMOBIL corpus, and a larger set makes consistent judgements harder. In addition, SWITCHBOARD consists of unstructured non-directed conversations, which contrast with the highly goal-directed dialogues of the VERBMOBIL corpus.

One approach that has been tried for dialogue act tagging is the use of n-gram language modelling, exploiting ideas drawn directly from speech recognition. For example, Reithinger & Klesen (1997) have applied such an approach to the VERBMOBIL corpus, which provides only a rather limited amount of training data, and report a tagging accuracy of 74.7%. Stolcke et al. (2000) apply

[2] See `http://www.dcs.shef.ac.uk/nlp/amities/`

a somewhat more complicated n-gram method to the SWITCHBOARD corpus (which employs both n-gram language models of individual utterances, and n-gram models over dialogue act sequences) and achieve a tagging accuracy of 71% on word transcripts, drawing on the full 205k utterances of the data. Of this, 198k utterances were used for training, with a 4k utterance test set. These performance differences can be seen to reflect the differential difficulty of tagging for the two corpora.

A second approach by Samuels et al. (1998), uses transformation-based learning over a number of utterance features, including utterance length, speaker turn and the dialogue act tags of adjacent utterances. They achieved an average score of 75.12% tagging accuracy over the VERBMOBIL corpus. A significant aspect of this work, that is of particular relevance here, has addressed the automatic identification of word sequences that would form dialogue act cues. A number of statistical criteria are applied to identify potentially useful n-grams which are then supplied to the transformation-based learning method to be treated as 'features'.

3.1 *Creating a naive classifier*

As noted, Samuels et al. (1998) investigated methods for identifying word n-grams that might serve as useful dialogue act cues for use as features in transformation-based learning. We decided to investigate how well n-grams could perform when used directly for dialogue act classification, i.e., with an utterance being classified solely from the individual cue phrases it contains. Two questions immediately arise. Firstly, which n-grams should be accepted as cue phrases for which dialogue acts, and secondly, which dialogue act tag should be assigned when an utterance contains several cues phrases that are indicative of different dialogue act classes. In the current work, we have answered both of these questions principally in terms of *predictivity*, i.e., the extent to which the presence of a certain n-gram in an utterance is predictive of it having a certain dialogue act category, which for an n-gram n and dialogue act category d corresponds to the conditional probability: $P(d\,|\,n)$.

A set of n-gram cue phrases was derived from the training data by collecting all n-grams of length 1–4, and counting their occurrences in the utterances of each dialogue act category and in total. These counts allow us to compute the above conditional probability for each n-gram and dialogue act. This set of n-grams is then reduced by applying thresholds of predictivity and occurrence, i.e., eliminating any n-gram whose maximal predictivity for any dialogue act falls below some minimum requirement, or whose maximal number of occurrences with any category falls below a threshold value. The n-grams that remain are used as cue phrases. The threshold values that were used in our experiments were arrived at empirically.

To classify an utterance, we identify all the cue phrases it contains, and determine which has the highest predictivity of some dialogue act category, and then that category is assigned. If multiple cue phrases share the same maximal predictivity, but predict different categories, one category is assigned arbitrarily. If no cue phrases are present, then a default tag is assigned, corresponding to the most frequent tag within the training corpus.

3.2 *Corpus, data sets and experiments*

For our experiments, we used the SWITCHBOARD corpus, which consists of 1,155 annotated conversations, comprising around 205k utterances. The dialogue act types for this set can be seen in Jurafsky et al. (1997). From this source, we derived two alternative datasets. Firstly, we extracted 50k utterances, and divided this into 10 subsets as a basis for 10-fold cross-validation (i.e., giving 45k/5k utterance set sizes for training/testing). This volume was selected as being large enough to give an idea of how well methods could perform where a good volume of data was available, but not too large to prohibit experiments with 10-fold cross-validation from excessive training times. The second data set was selected for loose comparability with the work of Samuel, Carberry and Vijay-Shanker on the VERBMOBIL corpus, who used training and test sets of around 3k and 300 utterances. Accordingly, we extracted 3300 utterances from SWITCHBOARD, and divided this for 10-fold cross-validation. We evaluated the naive tagging approach using these two data sets, in both cases using a predictivity threshold of 0.25 and an occurrence threshold of 8 to determine the set of cue phrases. Applied to the smaller data set, the approach yields a tagging accuracy of 51.8%, which compares against a baseline accuracy of 36.5% from applying the most frequently occurring tag in the SWITCHBOARD data set (which is **sd** — statement). Applied to the larger data set, the approach yields a tagging accuracy of 54.5%, which compares to 33.4% from using the most frequent tag.

Further experiments suggest that we can dramatically improve this score. We introduced start and end tags to every utterance (to capture phrases which serve as cues when specifically in these locations), and trained models sensitive to utterance length. For example, we trained three models — one for utterances of length 1, another for length between 2 and 4 words, and another for length 5 and above. Combining these features, we obtained a cross validated score for our naive tagger of 61.92% over the larger, 50k data set (with a high of 65.03%). Given that Stolke et al. achieve a total tagging accuracy of around 70% on SWITCHBOARD data, we observe that our approach goes a long way to reproducing the benefits of that approach, but using only a fraction of the data, and using a much simpler model (i.e., individual dialogue act cues, rather than a complete n-gram language model).

3.3 *Experiments with Transformation-Based Learning*

Transformation-Based Learning (TBL) was first applied to dialogue act modelling by Samuel, Carberry and Vijay-Shanker. They achieved overall scores of 75.12% tagging accuracy, using the VERBMOBIL corpus. As previously noted, an aspect of their work addressed the identification of potential cue phrases, for use as features during transformation based learning, i.e., so transformation rules can be learned which require the presence of a given cue as a context condition for the rule firing. In that work, the initial tagging state of the training data from which TBL learning would begin was produced by assigning every utterance a default tag corresponding to the most frequent tag over the entire corpus.

In our experiments, we wanted to investigate two issues. Firstly, whether a more effective dialogue act tagging approach could be produced by using our naive n-gram classifier as a pre-tagger generating the initial tagging state over which a TBL tagger could be trained. It seems plausible that the increased accuracy of the initial tag state produced by the naive classifier, as compared to assigning just the most frequent tag, might provide a basis for more effective subsequent training. Secondly, we wanted to assess the impact of using larger volumes of training data with a transformation based approach, given that Samuel et al.'s results are based on a quite small data set from the VERBMOBIL corpus.

For an implementation of transformation based learning, we used the freely available μ-TBL system of Lager (1999). The current distribution of μ-TBL provides an example system for dialogue act modelling, including a simple set of templates, which is developed with reference to the Samuel et al. work, and applied to the MapTask domain (Lager & Zinovjeva 1999). We have used this set of templates for all our experiments. We should note that the Lager and Samuel et al. template sets differ considerably, e.g., Samuel et al. use thousands of templates (together with a Monte Carlo variant of the training algorithm), whilst the μ-TBL templates are much fewer in number, may refer only to the dialogue act tags of preceding utterances (i.e., not both left and right), and may refer to any unigram or bigram appearing within utterances as part of a context condition, i.e., they are not provided with a fixed set of dialogue act cues to consider.

Our best results over the larger data set from the SWITCHBOARD corpus are around 66%, applying TBL to an initial data state produced by the naive classifier. Interestingly, our results indicate the naive classifier achieves most of the gain, with TBL consistently adding only 2 or 3%. In further work, we intend to apply other machine learning algorothms to the results of pre-tragging the data using a naive classifier.

3.4 *N-best dialogue act classification*

Our most recent experiment shows interesting promise. We built a classifier using the 45k utterance training set, and tested it on the 5k utterance test set. However, rather than attempting to find the single best match from the classifier, we tagged each utterance with the top 5 possible utterances, as indicated by the classifier on the basis of the predictivity of the n-grams the utterance contained. On a cross-validation of the corpus, we calculated that 86.74% of the time the correct dialogue act was contained in the 5-best output of the classifier. In order to create some baseline measure, this experiment was repeated using the top 5 n-grams occurring by frequency in the SWITCHBOARD corpus. The tagging accuracy of this experiment was 71.09%.

This would appear to confirm our belief in a limit on the potential resolution to this classification problem using n-gram based corpus analysis. However, we can offer an ordered list of possible alternatives to some higher level structure in the DM, where although the maximum attainable score is lower, the number of possible choices is reduced from 42 to 5.

4 Future work: Data driven dialogue discovery

Using the same corpus as above, to what extent could we discover the structure of DAFs, and their bounds from segmentations of the corpus, from annotated corpora? We are currently exploring the possibility of deriving the DAF structures themselves by taking a dialogue-act annotated corpus and then annotating it further with an information extraction engine (Larsson & Traum 2000) that seeks and semantically tags major entities and their verbal relations in the corpus, which is to say, derives a surface-level semantic structure for it. One function of this additional semantic tagging will be to add features for the DA tagging methods already described, in the hope of improving our earlier figures by adding semantic features to the learning process.

The other, and more original possibility, is that of seeking repeated sequences of DA amd semantic triple type (verb, plus agent and object types) and endeavouring to optimise the "packing" of such sequences to fill as much as possible of a dialogue by using some algorithm such as Minimum Description Length, so as to produce reusable, stereotypical, dialogue segments. We anticipate combining this with some corpus segmentation by topic alone, following Hearst's (1993) tiling technique.

Given any success at learning the segmentation of dialogue data, we expect to use some form of the Reinforcement Learning approach of Walker (1990) to optimise the DAF's themselves.

5 Discussion

The work in the last section is at a very preliminary stage, but will we hope form part of the general strategy of this paper which is to derive a set of weak, general, learning methods which will be strong in combination. This means the effect of the combination of a top down DAM using DAFs learned from corpus data, with a DA and semantics parser learned from the same data. It is the interaction of these bottom-up and top-down strategies in dialogue understanding (where the former is taken to include the ambiguities derived from the acoustic model) that we seek to investigate. This can perfectly well be seen as part of the program for a full dialogue model laid out in Young (2002) in which he envisaged a dialogue model as one where different parts are separately observed and framed before being combined.

We have shown that a simple dialogue act tagger can be created that uses just n-gram cues for classification. This naive tagger performs modestly, but still surprisingly well given its simplicity. More significantly, we have shown that a naive n-gram classifier can be used to pre-tag the input to tranformation based learning, which removes the need for a vast number of n-gram features to be used in the learning algorithm. One of the prime motivators for using TBL was its resiliance to such a high number of features, so by removing the need to incorporate them, we are hopeful that we can use a wider range of machine learning approaches for this task.

In regard to the naive n-gram classifier, we have described how the training of the classifier involves pruning the n-gram by applying thresholds for predictivity and absolute occurrence. These thresholds, which are empirically determined, are applied globally, and will have a greater impact in eliminating possible n-gram cues for the less frequently occurring dialogue act types. We aim to investigate the result of using local thresholds for each dialogue act type, in an attempt to keep a adequate n-gram representation of all dialogue acts types, including the less frequently occurring ones.

Finally, we aim to apply these techniques to a new corpus collected for the AMITIES project, consisting of human-human conversations recorded in the call centre domain (Hardy et al. 2002). We hope that the techniques outlined here will prove a useful first step in creating automatic service counters for call centre applications.

With the generation of more data from our already functioning DAM we hope to derive constraints on stack access and the reopening of all unpopped DAFs. This, if successful, will be an important demonstration of the different functioning of DAFs as contrasted with the use of ATNs in syntactic analysis (e.g., Woods 1970) where non-determinisn requires both back tracking and the exhaustion of all unpopped ATNs for completeness and the generation of all valid parsings of a sentence. It should be noted that this is not at all the case here: there is no pro-

vision for backtracking in DAFs and we expect to derive strong constraints such that not all unpopped DAFs will be reactivated. Analogous to the early dialogue findings of Grosz (1977) and Reichmann (1985) we expect some unpopped DAFs are not reopenable after substantial dialogue delay, just as they showed that dialogue segments and topics were closed off and became eventually inaccessible. Also, the ATN interpreter, unlike it's use in syntactic processing, is deterministic, since, in every state, there will be a best match between some arc condition and incoming representations. In this paper, we have discussed aspects of our approach to dialogue analysis/fusion and control, but have not touched at all on generation/fission and the role of knowledge rich items, such as belief and planning structures, in that phase.

Acknowledgements. This paper is based on work supported in part by the European Commission under the 5th Framework IST/HLT Program (consortia AMITIES and COMIC) and by the U.S. Defense Advanced Research Projects Agency.

REFERENCES

Allen, J. F., L. K. Schubert, G. Ferguson, P. Heeman, C. Hee Hwang, T. Kato, M. Light, N.G. Martin, B.W. Miller, M. Poesio & D.R. Traum. 1995. "The TRAINS Project: A Case Study in Building a Conversational Planning Agent". *Journal of Experimental and Theoretical AI* (*JETAI*) 7:7-48.

Grosz, Barbara. 1977. "The Representation and Use of Focus in Understanding Dialogs". *Readings in Natural Language Processing* ed. by Barbara Grosz, Karen Sparck Jones & Bonnie Lynn Webber, 353-362. Los Altos, Calif.: Morgan Kaufmann.

Hardy, H., K. Baker, L. Devillers, L. Lamel, S. Rosset, T. Strzalkowski, C. Ursu & N. Webb. 2002. "Multi-Layered Dialogue Annotation for Automated Multilingual Customer Service". *Proceedings of the ISLE workshop on Dialogue Tagging for Multimodal Human Computer Interaction*. Edinburgh.

Hardy, H., T. Strzalkowski & M. Wu. 2003. "Dialogue Management for an Automated Multilingual Call Center". *Proceedings of the HLT-NAACL 2003 Workshop on Research Directions in Dialogue Processing*, 10-12. Edmonton, Canada.

Hearst, Marti A. 1993. "TextTiling: A Quantitative Approach to Discourse Segmentation". Technical Report UCB:S2K-93-24. Berkeley, Calif.: Univ. of California.

Jurafsky, D., R. Bates, N. Coccaro, R. Martin, M. Meeter, K. Ries, E. Shriberg, A. Stolcke, P. Taylor & C. Van Ess-Dykema. 1998. "Switchboard Discourse Language Modeling". Project Report Research Note 30. Baltimore, Maryland: Center for Speech and Language Processing, Johns Hopkins University.

Jurafsky, D., E. Shriberg & D. Biasca. 1997. "Switchboard-DAMSL Labeling Project Coder's Manual". Technical Report 97-02. Boulder, Colorado: Institute of Cognitive Science, Univ. of Colorado.

Lager, T. 1999. "The μ-TBL System: Logic Programming Tools for Transformation-Based Learning". *Proceedings of the 3rd International Workshop on Computational*

Natural Language Learning (*CoNLL'99*). Bergen, Norway. — `http://cnts.uia.ac.be/conll99/programme.html` [Source checked in May 2004]

Lager, T. & N. Zinovjeva. 1999. "Training a Dialogue Act Tagger with the μ-TBL System". *Proceedings of the 3rd Swedish Symposium on Mulitmodal Communication*. Linköping University Natural Language Processing Laboratory. — `http://www.ida.liu.se/~ssomc/proceedings.html` [Source checked in May 2004]

Larsson, S. & D. Traum. 2000. "Information State and Dialogue Management in the TRINDI Dialogue Move Engine Toolkit". *Best Practice in Spoken Language Dialogue Systems Engineering* (= Special Issue of *Natural Language Engineering*, 6:3/4), 323-340.

Lemon, Oliver, A. Bracy, A.R. Gruenstein & S. Peters. 2001. "The Witas Multi-Modal Dialogue System I". *Proceedings of the 7th European Conference on Speech and Communication Technology* (*EuroSpeech 2001*), 1559-1562. Aalborg, Denmark.

Reichmann, R. 1985. *Getting Computers to Talk Like You and Me*. Cambridge Mass.: MIT Press.

Reithinger, N. & M. Klesen. 1997. "Dialogue Act Classification Using Language Models". *Proceedings of the 5th European Conference on Speech and Communication Technology* (*EuroSpeech 1997*), 2235-2238. Rhodes, Greece.

Reithinger, N. & E. Maier. 1995. "Utilizing Statistical Dialogue Act Processing in Verbmobil". *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics* (*ACL-95*), 116-121. MIT, Cambridge, Massachusetts.

Samuel, K., S. Carberry & K. Vijay-Shanker. 1998. "Dialogue Act Tagging with Transformation-Based Learning". *36th Annual Meeting of the Association for Computational Linguistics and 17th Annual International Conference on Computational Linguistics* (*ACL-COLING'98*), vol.2, 1150-1156. Montréal, Canada.

Samuel, K., S. Carberry & K. Vijay-Shanker. 1999. "Automatically Selecting Useful Phrases for Dialogue Act Tagging". *Proceedings of the 4th Conference of the Pacific Association for Computational Linguistics*. Waterloo, Ontario, Canada. — `http://www.cs.toronto.edu/~bowen/pacling/index.html` [Source checked in May 2004]

Stolcke, A., K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. Van Ess-Dykema & M. Meteer. 2000. "Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech". *Computational Linguistics* 26:3.339-373.

Walker, M.A. 1990. "An Application of Reinforcement Learning to Dialogue Strategy Selection in a Spoken Dialogue System for Email". *Journal of Artificial Intelligence Research* 12:387-416.

Woods, W.A. 1970. "Transition Network Grammars for Natural Language Analysis". *Communications of the ACM*, 13:10.591-606.

Young, S.J. 2000. "Probabilistic Methods in Spoken Dialogue Systems". *Philosophical Transactions of the Royal Society* (= *Series A*) 358:1769.1389-1402.

Learning Domain Theories

STEPHEN G. PULMAN & MARIA LIAKATA
Computational Linguistics Group, Centre for Linguistics, Oxford University

Abstract

By a 'domain theory' we mean a collection of facts and generalisations or rules which capture what commonly happens (or does not happen) in some domain of interest. As language users, we implicitly draw on such theories in various disambiguation tasks, such as anaphora resolution and prepositional phrase attachment, and formal encodings of domain theories can be used for this purpose in natural language processing. They may also be objects of interest in their own right, that is, as the output of a knowledge discovery process. We describe a method of automatically learning domain theories from parsed corpora of sentences from the relevant domain.

1 Domain theories

The resolution of ambiguity has been a perennial topic of interest among linguists and computational linguists. In the early days of generative linguistics, the existence of various types of ambiguity was used to justify abstract levels of linguistic structure:

(1) Flying planes can be dangerous.
(2) He saw the man with the telescope.

These examples show, respectively, that the same sequence of words can have different assignments of parts of speech (lexical categories), and that the same sequence of parts of speech can have differing constituent structures associated with them. However, after some initial flirtations with 'semantic features', the problem of resolving such ambiguities in a given context was not seriously pursued, since it was regarded as what we would now call 'AI-complete', requiring the encoding of salient features of the context as well as vast amounts of non-linguistic knowledge:

> For practically any item of information about the world, the reader will find it a relatively easy matter to construct an ambiguous sentence whose resolution requires the representation of that item (Katz & Fodor, 1964:489).

Early work in computational linguistics reinforced this conclusion, pointing out that it is not just lists of facts that are involved in disambiguation, but reasoning based on those facts. Furthermore, the kind of contextual reasoning required to disambiguate some sentences can draw on facts which require a subtle understanding of social or political structures, rather than facts about the physical world:

(3) The police refused the students permission to demonstrate because they advocated/feared violence.

The decision as to whether 'they' refers to 'the police' or to 'the students' with one or another of the alternative verbs, seems to be given by our assessment of the relative plausibilities of the propositions 'police/students advocate violence' and 'police/students fear violence', and the likelihood that each of these propositions could form a reason for the police withholding permission. This decision involves judging the plausibility of a proposition inferred from things already known, but itself novel: it is very unlikely that a hearer has consciously or unconsciously worried on any previous occasion about whether police advocate or fear violence more than students do.

Current implemented approaches to the disambiguation problem rely on statistical methods, for the most part. Starting with a corpus of disambiguated sentences (a treebank), some kind of classifier is trained to distinguish 'good' from 'bad' parses of new sentences. The classifier will be trained on 'features' in the machine learning sense, typically some combination of head words and grammatical or structural relations. This classifier may be implicit as an integral part of the parsing algorithm (e.g., Collins 1997) or a separate component acting as a ranking mechanism on the output of an N-best parser (as in early work like Alshawi and Carter 1994, or more recent systems like Collins and Duffy 2002).

At the current state of the art, training a statistical classifier to perform disambiguation is the method of choice, and will deliver a level of performance that traditional symbolic methods, even where they are available, are unable to approach. Nevertheless, our position is that while these techniques are clearly the best current engineering solution to the problem of disambiguation, they are unsatisfactory in several scientific respects. Firstly, these solutions do not easily (or at all) generalise from one type of disambiguation problem to another. For example, in the following sentences it is intuitively clear that the same piece of general knowledge - that you can cut bread with a knife - is used to carry out a PP attachment decision, interpret a compound nominal, and assign an antecedent to a pronoun:

(4) Tell him to cut the bread with the knife.

(5) He picked up the bread knife.

(6) Put the bread on the wooden board. Use the knife to slice it.

But it is not at all straightforward to reuse the type of classifier that one might build to disambiguate the PP attachment for the remaining two interpretation tasks.

Secondly, it is by no means clear that one can in practice extend statistical methods to the case of context-dependent disambiguation, or to the kind of reasoning found in cases of conversational implicature or 'relevance'. Many disambiguation decisions require knowledge of what entities are salient, unique, etc.

Consider the task of deciding whether 'by the factory' should be construed as an agentive or a locative phrase in the following examples (taken from a Google search for 'by the factory'):

(7) Four (4) Whelen #64 strobe lights shall be located on the rear of the body, over the rear taillights. The lens colors shall be (2) red, (1) amber, and (1) blue. They shall be strategically located by the factory using good judgement.

(8) In our offices, located by the factory, we have our export department where experienced pesonnel attends the requirements of our export costumers *(sic)* ...

The correct decision in the first example requires the correct pronoun resolution to have been made. Surely the task of getting enough context and disambiguated example training material pairs would be in practice impossible: the sparse data problem is bad enough for context-independent sentence disambiguation.

What is the alternative? We suggest that the traditional wisdom — that you need a lot of knowledge of the world to make disambiguation decisions — is largely correct. Furthermore, the observation that this knowledge of the world is not just a list of facts, but is organised in a deductive structure that allows you to make novel inferences is also largely correct. The question is, how do we get such theories? The early pessimism about the possibility of building large scale monolithic theories by hand is surely justified, although there have been some heroic (and expensive) attempts (Lenat 1995). Moreover, it is not clear that a single 'theory of the world' is what is wanted: experience in using general linguistic classification schemes, such as WordNet (Fellbaum 1998), suggests that in particular domains, word usage and relationships can be rather different from 'general usage', which is presumably based on an abstraction from many such domains, not necessarily chosen systematically. Rather, a more focused and less ambitious aim is to develop a theory for a particular domain: by 'domain theory' we mean a collection of facts and generalisations or rules which capture what commonly happens (or does not happen) in some domain of interest.

Domain specific hand-crafted theories have been developed from time to time: the earliest was perhaps embodied in the 'preference semantics' of Wilks (1975) (although this was in fact quite general in its intended coverage) and more recent attempts have been described by Hobbs et al. (1993). However, even constructing such smaller theories is still an extremely labour intensive business.

What we need is some way of automatically, or semi-automatically, inducing domain theories from data of some kind. But from what kind of data? Here, we can perhaps go back to the original observation that in order to make a disambiguation decision, we need a great deal of knowledge about the world. However, the observation that disambiguation decisions depend on knowledge of the world can be made to cut both ways: *just as we need a lot of knowledge of the world*

to make disambiguation decisions, so a given disambiguation decision can be interpreted as telling us a lot about the way we view the structure of the world. Since in the general case it is a much easier job to disambiguate sentences than to directly encode the theory that we are drawing on in so doing, a better strategy for trying to build a domain theory would be to try to capitalise on the information that is tacitly contained in those disambiguation decisions.

2 A partial ATIS domain theory

In Pulman (2000) it was shown how it was possible to learn a simple domain theory from a disambiguated corpus: a subset of the ATIS (air travel information service) corpus (Godfrey & Doddington 1990). Ambiguous sentences were annotated as shown to indicate the preferred reading:

```
[i,would,like,
   [the,cheapest,flight,from,washington,to,atlanta]]

[can,i,[have,a,steak_dinner],on,the,flight]

[do,they,[serve,a,meal],on,
         [the,flight,from,san_francisco,to,atlanta]]

[i,would,like,[a,flight,from,boston,to,san_francisco,
                  [that,leaves,before,'8:00']]]
```

The 'good' and the 'bad' parses were used to produce simplified first order logical forms representing the semantic content of the various readings of the sentences. The 'good' readings were used as positive evidence, and the 'bad' readings (or more accurately, the bad parts of some of the readings) were used as negative evidence. For example, the good and bad readings of the first example above are:

Good:

```
∃A.flight(A) & from(A,washington) & to(A,atlanta)
  & cheapest(A) & would_like(e73,I,A)
```

Bad:

```
∃A.flight(A) & from(A,washington) & cheapest(A)
  & would_like(e75,I,A) & to(e75,atlanta)
```

We use an eventish semantics for verbs: 'e73' etc are skolem constants deriving from an event quantification, denoting events. The bad reading claims that it is the liking event that is going to Atlanta, not the flight. Note that some components of the logical form are common to both the good and bad readings: when shared components are factored out, and we have skolemised and transformed to clausal form, the positive and negative evidence derived from this sentence would be:

Positive:

```
cheapest(sk80).
flight(sk80).
from(sk80,washington).
to(sk80,atlanta).
would_like(e73,I,sk80).
event(e73).
```

Negative:

```
not(to(e73,atlanta))
```

Note that we have added some extra background sortal information to distinguish events from other individuals. We also add some other background sortal information: that United, American etc. are airlines, that Atlanta and Washington are cities, and so on.

Next we used a particular Inductive Logic Programming algorithm, Progol (Muggleton 1994), to learn a theory of prepositional relations in this domain: i.e., what kinds of entities can be in these relations, and which cannot:

```
on(+any,+any)
from(+any,+any)
to(+any,+any)
at(+any,+any)
```

The +any declaration says that we have no prior assumptions about sortal restrictions on these predicates. Among others we learn generalisations like these (all variables are implicitly universally quantified):

```
fare(A)   & airline(B) → on(A,B)
event(A)  & flight(B)  → on(A,B)
meal(A)   & flight(B)  → on(A,B)
flight(A) & day(B)     → on(A,B)
flight(A) & airline(B) → on(A,B)
```

Fares and flights are on airlines; events (like serving a meal) can be on flights, meals can be on flights, and flights can be on (particular) days. However:

```
event(A) & airline(B)  → not(on(A,B))
event(A) & city(B)     → not(from(A,B))
event(A) & city(B)     → not(to(A,B))
```

— events cannot be on airlines, and events don't go to or from cities. Using a different ILP algorithm, WARMR (Dehaspe & De Raedt 1997), for discovering frequent patterns (in data mining terms, association rules) and the system Aleph (Srinivasan 2003) for deriving constraints from these rules, we obtained the following facts:

Flights are direct or return, but not both:

```
direct(A)              → flight(A)
return(A)              → flight(A)
direct(A) & return(A)  → false
```

These generalisations are true with respect to the given corpus, but may not be true more generally. However, these constraints between prepositions are presumably of general validity:

```
from(A,B)  & to(A,B)     → false
at(A,B)    & after(A,B)  → false
at(A,B)    & before(A,B) → false
after(A,B) & before(A,B) → false
etc.
```

Having learned this domain theory we were able to show that it could be used successfully in disambiguating a small held-out section of the corpus, by checking for consistency between logical forms and domain theories. There are many variations to be explored here, but in this particularly simple setting the negative domain theory, i.e., the generalisations or constraints about what does NOT happen are the easiest to employ. A 'good' reading of an ambiguous sentence will be consistent with a negative constraint, whereas a 'bad' reading will lead to a contradiction. This can be implemented using a model builder for first order logic, in our case, a version of Satchmo (Manthey & Bry 1988). We assume as axioms the negative domain theory. We parse a sentence, and turn each parse tree into a first order logical form. We then add each logical form in turn to the axioms, and see whether we can build a model, i.e., whether the conjunction of the negative domain theory and the logical form is satisfiable or consistent. If we can build a model, the logical form is a plausible one; if we cannot, it is an implausible one. To illustrate from the example of a bad logical form given earlier:

```
∃A.flight(A) & from(A,washington)
  & cheapest(A) & would_like(e75,I,A)
  & to(e75,atlanta)
  ...& event(e75) & city(atlanta)
```

will contradict:

```
∀A,B.event(A) & city(B) → not(to(A,B))
```

and so no model can be constructed.

While the numbers of sentences involved in this experiment are too small for the results to be statistically meaningful, the experiment proved that the method works in principle, although of course in reality the notion of logical consistency is too strong a test in many cases. Note also that the results of the theory induction process are perfectly comprehensible - we get a theory with some logical structure, rather than a black box probability distribution. The theory we have got can be edited or augmented by hand, and could be used for other types of disambiguation task, such as word disambiguation and pronoun resolution. It could also be used for other types of NLP task altogether: logical domain theories have recently been used to improve performance in natural language question answering tasks (Moldovan et al 2003) and it has been argued that automatic creation of templates for Information Extraction could benefit from such domain theories

(Collier 1998). The output of our theory induction process thus meets the criteria of transparency and reusability discussed earlier.

3 Scaling up

Of course, the ATIS corpus is relatively simple and homogeneous. The question is whether this technique will scale up to larger and noisier corpora. However, this is rather difficult to do in practice since the method requires the corpus in question to be analysable to the extent that complete logical forms can be recovered for both good and bad readings of sentences. We know of no such large corpora: however, the Penn Tree Bank(Marcus et al. 1994) is a good starting point, since the syntactic annotations for sentences given there are intended to be complete enough for semantic interpretation, in principle, at least.

In practice, as we reported in Liakata and Pulman (2002), it is by no means easy to do this. We were able to recover partial logical forms from a large proportion of the treebank, but these are not complete or accurate enough to simply replicate the ATIS experiment. In order to approach this we selected about 40 texts containing the verb 'resign', all reporting, among other things, 'company succession' events, a scenario familiar from the Message Understanding Conference (MUC) task (Grishman & Sundheim 1996). The texts amounted to almost 4000 words in all. Then we corrected and completed the automatically produced logical forms by hand to get a fairly full representation of the meanings of these texts (as far as is possible in first order logic). We also resolved by hand some of the simpler forms of anaphoric reference to individuals to simulate a fuller discourse processing of the texts.

To give an example, a sequence of sentences like:

> J.P. Bolduc, vice chairman of W.R. Grace & Co. ... was elected a director. He succeeds Terrence D. Daniels,... who resigned.

was represented by the following sequence of literals:

```
verb(e1,elect).
funct_of('J.P._Bolduc',x1).
...
subj(e1,unspecified).
obj(e1,x1).
description(e1,x1,director,de1).


verb(e5,succeed).
subj(e5,x1).
funct_of('Terrence_D._Daniels',x6).
obj(e5,x6).
verb(e4,resign).
subj(e4,x6).
```

The representation is a little opaque, for various implementation reasons. It can be paraphrased as follows: there is an event, e1, of electing, the subject of which is unspecified, and the object of which is x1. x1 is characterised as 'J P Bolduc', and e1 assigns the description de1 of 'director' to x1. There is an event e5 of succeeding, and x1 is the subject of that event. The object of e5 is x6, which is characterised as Terrence D Daniels. There is an event e4 of resigning and the subject of that event is x6.

The reason for all this logical circumlocution is that we are trying to learn a theory of the 'verb' predicate, in particular we are interested in relations between the arguments of different verbs, since these may well be indicative of causal or other regularities that should be captured in the theory of the company succession domain. If the individual verbs were represented as predicates rather than arguments of a 'verb' predicate we would not be able to generalise over them: we are restricted to first order logic, and this would require higher order variables.

We also need to add some background knowledge. We assume a fairly simple flat ontology so as to be able to reuse existing resources. Some entities were assigned to classes automatically (see below for details of the clustering techniques used), others had to be done by hand. The set of categories used were:

> *company, financial instrument, financial transaction, location, money, number, person, company position, product, time, and unit (of organisation).*

As before, the representation has these categories as an argument of a 'class' predicate to enable generalisation:

```
class(person,x1).
class(company,x3).
etc.
```

Ideally, to narrow down the hypothesis space for ILP, we need some negative evidence. In the ATIS case, we were able to do this because our parser was able to find all parses for the sentences in question, good and bad. In the case of the Penn Tree Bank, only the good parse is represented. There are several possible ways of obtaining negative data, of course: one could use a parser trained on the Tree Bank to reparse sentences and recover all the parses. However, there still remains the problem of recovering logical forms from 'bad' parses. An alternative would be to use a kind of 'closed world' assumption: take the set of predicates and arguments in the good logical forms, and assume that any combination not observed is actually impossible. One could generate artificial negative evidence this way.

Alternatively, one can try learning from positive only data. Progol and Aleph are able to learn from positive only data, with the appropriate settings. Also, so-called 'descriptive' ILP systems like WARMR do not always need negative data: they are in effect data mining engines for first order logic, learning generalisations and correlations in some set of data.

4 Domain theory for company succession events

We found that the most successful method, given the absence of negative data, was to use WARMR to learn association rules from the positive data. As with all types of association rule learning, WARMR produces a huge number of rules, of varying degrees of coverage. We spent some time writing filters to narrow down the output to something useful. Such filters consist of constraints ruling out patterns that are definitely not useful, for example patterns containing a verb but no arguments or attributes. An example of such a restriction is provided below:

```
pattern_constraint(Patt):-
     member(verb(_,E,_A,_,_),Patt),
     (member(attr(_,E,Attr),Patt)
       ->
      \+constraint_on_attr(Patt,Attr)).
```

This says that a rule isn't useful unless it contains a verb and one of its attributes that satisfies a certain constraint. A constraint is of the following form:

```
constraint_on_attr(Patt, Attr) :-
      member(class(_,Attr), Patt).
```

The above states that there should be a classification of the attribute Attr present in the rule. A useful pattern Patt will satisfy such constraints and thus make the 'pattern_constraint' predicate fail.

Some of the filtered output, represented in a more readable form compatible with the examples above are:

Companies report financial transactions:

```
subj(B,C) & obj(B,D) & class(fin_tran,D) & class(company,C)
  → verb(B,report)
```

Companies acquire companies:

```
subj(B,C) & obj(B,D) & class(company,D) & class(company,C)
  → verb(B,acquire)
```

Companies are based in locations:

```
obj(A,C) & class(company,C) & in(A,D) & class(location,D)
  → verb(A,base)
```

If a person is elected, another person resigns:

```
verb(H,elect) & obj(H,I) & class(person,I) & subj(C,L)
& class(person,L)
  → verb(C,resign)
```

If person C succeeds person E, then someone has elected person C:

```
obj(A,C) & class(person,C) & verb(D,succeed) & subj(D,C)
& obj(D,E) & class(person,E)
  → verb(A,elect)
```

If someone elects person C, and person D resigns, then C succeeds D:

```
subj(G,C) & verb(A,elect) & obj(A,C) & class(person,C)
& verb(E,resign) & subj(E,D) & class(person,D)
   → verb(G,succeed)
```

While there are many other rules learned that are less informative than this, the samples given here are true generalisations about the type of events described in these texts: unremarkable, perhaps, but characteristic of the domain. It is noteworthy that some of them at least are very reminiscent of the kind of templates constructed for Information Extraction in this domain, suggesting a possible further use for the methods of theory induction described here.

In the ATIS domain, we were able to evaluate the usefulness and accuracy of the rules induced by applying them in a syntactic disambiguation task. However, we have not been able to replicate that evaluation here, for a variety of reasons of which the most important is the relative sparseness of the rules derived: much more information would be needed in most cases for successful disambiguation.

5 Next steps

In our current work we are trying to scale up again. However, to do this successfully we need better ways of obtaining negative evidence, and of formalising background knowledge. For the former, we intend to experiment with a wide coverage parser which is capable of delivering all analyses for sentences rather than simply the best one. For the latter we have used clustering techniques to try to group logical predicates into classes which will allow the various ILP algorithms to generalise over samples of evidence involving these predicates. In the ATIS experiment we carried this step out by hand, whereas for the company succession succession events we used a mixture of automatically derived and hand made classes. Obviously, a larger data set would require full automation of the process.

In our current experiments we have tried clustering the entire set of logical form verb predicates derived from the PTB. More specifically, we clustered the verb argument slots of each predicate (e.g., resign_arg1 is the subject of the verb resign). Note that the logical form predicates are derived automatically from the stem form of the word involved, and so no word sense disambiguation is being carried out, leading to a major source of noise in the data. We constructed a large three-dimensional matrix:

predicates X arguments X frequency

An artificial example of such a matrix is:

	cat	dog	father	computer
snore_arg1	3	2	10	0
hit_arg1	2	4	1	2
hit_arg2	6	7	2	20

This says that *cat* occurs as first argument of 'snore' three times, *dog* twice, *father* ten times and *computer* not at all. We used Autoclass (Cheeseman 1995) to form clusters of verb argument slots such that the ones grouped together are those verb slots filled by the same words. Classification of verbs and words that feature as their arguments was then easily derivable.

We ran Autoclass assuming that word frequencies follow a normal distribution, with dependencies between members of the same class. Autoclass found between 32 to 55 clusters, depending on whether the table consisted of absolute frequencies, logarithms of frequencies or whether there had been a pre-processing stage for grouping together person names, locations, numerical expressions and company names. This pre-processing stage involved checking against various gazetteer lists. We decided to stick with 32 clusters, since the classification into more groups did not make the intuitive basis for the clusters any more obvious. These were the clusters obtained for the absolute frequencies of the pre-processed data.

Many of the resulting classes are difficult to interpret, although about half have some clear intuitive basis, once some outliers are ignored:

```
Class 9:  plunge, soar, climb, decline, jump, slide,
          plummet, pick up, slip
```

These verbs seem to have a common theme of a sudden movement, and when we look at the most frequently occurring arguments of some of them it becomes clear that they describe sudden movements of what might be called the class of 'financial indicators':

```
PLUNGE_arg1: subject argument:
  dow_jones_industrial_average, income, stock, person,
  profit, market, earnings, average, net,
  share, price

SOAR_arg1: subject argument:
  rate, price, profit, location, cost, dollar,
  volume, income, asset, interest, circulation,
  stock, revenue, jaguar_shares, earnings, person

PLUMMET_arg1:
  price, market, stock, ual_stock, earnings, profit,
  company, indicator, yield

JUMP_arg1:
  income,  profit, price, person, company, revenue,
  location, earnings, index, dow_jones_industrial_average,
  contract, yield, cost, gold, people
```

This class of verbs seems to involve a causal mechanism for a similar type of financial change:

```
Class 10:  bolster, increase, reduce, boost, occur, trigger
```

We find a class of aspectual verbs:

```
Class 16:  close, begin, start, complete
```

and we also find the class of the 'company succession verbs'

```
Class 18:  appoint, resign, succeed
```

Some other classes also have a clear semantic relationship to each other:

```
Class 23:  know, believe, think
Class 24:  buy, pay, purchase, own
```

As well as Autoclass, which is essentially classification by Expectation Maximisation, we also tried another method of clustering based on Independent Component Analysis (Roberts & Everson 2001). This latter approach involved obtaining the matrix of frequencies and minimizing its dimensions. Four classes resulted from this method but while easily interpretable, they were too general to be of use for our purposes. Nevertheless, the theoretical foundation of this clustering technique may well make it more suitable for future work, since it makes more realistic assumptions about the distributions our samples are from.

In principle, the way we would proceed having satisfactorily grouped predicates into clusters (perhaps hierarchical, although so far we have not attempted this) would be then to assign informative labels to clusters, and represent the relations between clusters and their members as background knowledge:

```
plunge(X)                    → move_suddenly(X)
soar(X)                      → move_suddenly(X)
plummet(X)                   → move_suddenly(X)
...
financial_indicator(X)       → plummet(X)
financial_indicator(X)       → soar(X)
...
dow_jones_industrial_avge(X) → financial_indicator(X)
earnings(X)                  → financial_indicator(X)
market(X)                    → financial_indicator(X)
...
```

This would enable the learning algorithm to generalise correctly over related examples.

As for evaluation, our original hope was to be able to use the derived theory both for syntactic disambiguation and for other tasks like reference resolution. In particular, we wanted to be able to deal with the cases that currently successful pronoun resolution algorithms (e.g., Hobbs 1978, Boguraev and Kennedy 1996) get wrong. These are cases where the structural configurations preferred by these algorithms yield interpretations that to us are clearly implausible. Lappin (2004) has argued that however successful a structural or statistically based algorithm is, there will always be a residue of cases like these where detailed world knowledge and reasoning is required to get the right result.

The following examples from the Penn Tree Bank illustrate the phenomenon in question:

(9) The government$_i$ counts money as it$_i$ is spent.

Parallelism and grammatical relation salience will combine to strongly prefer the reading indicated, which is of course wrong. In order to override this reading and select the correct one you need to know that while it is possible for money to be spent, it is not possible for a government to be spent (at least, not in the same sense).

An example close to the kind of clusters we have been able to induce is this one, where again the structural preference would be as indicated:

(10) Investors$_i$ usually notice markets because they$_i$ rise or fall rapidly.

This interpretation requires it to be more plausible that investors rise or fall than that markets do. We would hope that the correct preference would be captured by generalisations like those sketched above.

It is arguable that a statistical system for pronoun resolution, which infers sortal restrictions on arguments from collocations, would probably be able to deal with the above example. However, this is certainly not the case for examples like the following:

(11) Leaders$_i$ of several Middle Eastern terrorist organisations met in Teheran to plan terrorist acts. Among them$_i$ was the PFL of Palestine...

Again, if we ignore the content of the sentence, the most likely antecedent for 'them' is the one indicated. In order to see that this is incorrect, we need to know an axiom associated with 'among': roughly, that if X is among Ys, and all Ys are P, then X is P. However, this axiom is somewhat more abstract than those we are learning (although domain specific versions are attainable) and it remains to be seen whether it is possible to learn a theory with enough width and depth of coverage to be properly evaluated on something like a pronoun disambiguation task.

A further possibility is to use the domain theory learning method described here to learn *new* knowledge. The domain theories we have described so far are trying to encapsulate the 'common-sense' understanding that is required to interpret and disambiguate texts - this is routine, everyday knowledge that members of the relevant community can be expected to share. But if we instead had parsed texts from some technical domain - for example, the protein interaction research abstracts used in the information extraction experiment described in Thomas et al. 2000 - it is conceivable that an induced theory might well lead one to be able to deduce some completely new and useful information about that domain. By putting together information from different texts in the form of facts and inference rules, it may be possible to derive new conclusions that would not otherwise have been apparent.

Acknowledgements. We have had a great deal of help from members of the ILP community. We would particularly like to thank Ashwin Srinivasan (IBM, New Delhi), Steve

Moyle (Oxford) and James Cussens (York) for their help with Aleph and Jan Struyf, Hendrik Blockeel and Jan Ramon (K.U. Leuven), for their generous help with WARMR. We also thank Steve Roberts (Oxford) for advice on clustering and for carrying out the ICA analysis for us.

REFERENCES

Alshawi, H. & D. Carter. 1994. "Training and Scaling Preference Functions for Disambiguation". *Computational Linguistics* 20:4.635-648.

Boguraev, Bran & C. Kennedy. 1996. "Anaphora for Everyone: Pronominal Anaphora Resolution without a Parser". *Proceedings of the 17th International Conference on Computational Linguistics* (*COLING 1996*), 113-118. Copenhagen, Denmark.

Cheeseman, P., J. Kelly, M. Self, J. Stutz, W. Taylor, & D. Freeman. 1998. "AutoClass: A Bayesian Classification System". *Proceedings of the 5th International Conference on Machine Learning, Ann Arbor, Michigan, June 12-14*, 54-64. San Francisco, Calif.: Morgan Kaufmann.

Collier, Robin. 1996. "Automatic Template Creation for Information Extraction, An Overview". Technical Report CS-96-07. Sheffield, U.K.: University of Sheffield, Department of Computer Science.

Collier, Robin. 1998. "Automatic Template Creation for Information Extraction". Ph.D. dissertation, University of Sheffield. Sheffield, U.K.

Collins, Michael & Nigel Duffy. 2002. "New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron". *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (*ACL'02*), 263-270. Philadelphia, Pennsylvania, U.S.A.

Collins, Michael. 1997. "Three Generative, Lexicalised Models for Statistical Parsing". *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics* (*jointly with the 8th Conference of the European Association for Computational Linguistics*), 16-23. Madrid, Spain.

Dehaspe, L. & L. De Raedt. 1997. "Mining Association Rules in Multiple Relations". *Proceedings of the 7th International Workshop on Inductive Logic Programming* (= *Lecture Notes in Artificial Intelligence*, 1297), 125-132. Berlin: Springer-Verlag.

Dehaspe, L. 1998. "Frequent Pattern Discovery in First-Order Logic". Ph.D. dissertation, Katholieke Universiteit Leuven. Leuven, Belgium.

Fellbaum, Christiane, ed. 1998. *WordNet: An Electronic Lexical Database*. Cambridge, Massachusetts: MIT Press.

Hemphill, C., John Godfrey & George Doddington. 1990. "The ATIS Spoken Language Systems Pilot Corpus". *Speech and Natural Language Workshop, Hidden Valley, Pennsylvania*, 96-10. San Francisco, Calif.: Morgan Kaufmann.

Grishman, Ralph & Beth Sundheim. 1996. "Message Understanding Conference-6: A Brief History". *Proceedings of the 16th International Conference on Computational Linguistics* (*COLING'96*), 466-471. Copenhagen, Denmark. — `www.cs.nyu.`

edu/cs/projects/proteus/muc/muc6-history-coling.ps [Source checked in May 2004]

Hobbs, Jerry R. 1978. "Resolving Pronoun Preferences". *Lingua* 44.311-338.

Hobbs, Jerry R., Mark E. Stickel, Douglas Appelt & Paul Martin. 1993. "Interpretation as Abduction". *Artificial Intelligence* 63:1/2.69-142.

Katz, Jerrold J. & Jerry A. Fodor. 1964. "The Structure of a Semantic Theory." *The Structure of Language: Readings in the Philosophy of Language* ed. by J.J. Katz & J.A. Fodor, 479-518. Englewood Cliffs, N.J.: Prentice Hall.

Lappin, Shalom & Herbert J. Leass. 1993. "An algorithm for Pronominal Anaphora Resolution". *Computational Linguistics* 20:4.535-561.

Lappin, Shalom. 2004. "A Sequenced Model of Anaphora and Ellipsis Resolution". *Anaphora Processing: Linguistic, Cognitive, and Computational Modelling* ed. by A. Branco, A. McEnery & R. Mitkov. Amsterdam: John Benjamins.

Lenat, Douglas B. 1995. "CYC: A Large-scale Investment in Knowledge Infrastructure". *Communications of the Association for Computing Machinery* (*CACM*) 38:11.33-38.

Liakata, Maria & Stephen G. Pulman. 2002. "From Trees To Predicate-Argument Structures". *Proceedings of the 20th International Conference on Computational Linguistics* (*COLING 2002*), 563-569. Taipei, Taiwan.

Manthey, Rainer & Francois Bry. 1988. "SATCHMO: A Theorem Prover Implemented in Prolog". *Proceedings of the 9th International Conference on Automated Deduction, Argonne, Illinois, U.S.A.* ed. by Ewin L. Lusk & Ross A. Overbeek (= *Lecture Notes In Computer Science* LNCS, 310), 415-434. Berlin: Springer.

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz & Britta Schasberger. 1994. "The Penn Treebank: Annotating Predicate Argument Structure". *ARPA Human Language Technology Workshop*. Princeton, N.J.

Moldovan, Dan, C. Clark, S. Harabagiu & S. Maiorano. 2003. "COGEX: A Logic Prover for Question Answering". *Proceedings of Human Language Technology-North American Association for Computational Linguistics* (*HLT-NAACL'03*), 87-93. Edmonton, Canada.

Muggleton, S. 1995. "Inverse Entailment and Progol". *New Generation Computing* vol.13, 245-286.

Pulman, Stephen G. 2000. "Statistical and Logical Reasoning in Disambiguation". *Philosophical Transactions of the Royal Society of London*, Series A, vol.358, 1267-1280.

Roberts, S. & R. Everson. 2001. *Independent Component Analysis: Principles and Practice*. Cambridge: Cambridge University Press.

Srinivasan, Ashwin. 2003. "The Aleph Manual". Oxford: Oxford University, Machine Learning Laboratory. — www.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/ [Source checked in May 2004]

Thomas, J., D. Milward, C. Ouzounis, S.G. Pulman, & M. Carroll. 2000. "Automatic Extraction of Protein Interactions from Scientific Abstracts". *Proceedings of the Pacific Symposium on Biocomputing 2000*, 541-552. Hawaii.

Wilks, Yorick. 1975. "Preference Semantics". *Formal Semantics of Natural Language*, ed. by E.L. Keenan, 329-348. Cambridge: Cambridge University Press.

Recent Developments in Temporal Information Extraction

INDERJEET MANI

Georgetown University

Abstract

The growing interest in practical NLP applications such as text summarization and question-answering places increasing demands on the processing of temporal information in natural languages. To support this, several new capabilities have emerged. These include the ability to tag events and time expressions, to temporally anchor and order events, and to build models of the temporal structure of discourse. This paper describes some of the techniques and the further challenges that arise.

1 Introduction

Natural language processing has seen many advances in recent years. Problems such as morphological analysis, part-of-speech tagging, named entity extraction, and robust parsing have been addressed in substantial ways. Hybrid systems that integrate statistical and symbolic methods have proved to be successful in particular applications. Among the many problems remaining to be addressed are those that require a deeper interpretation of meaning. Here the challenges in acquiring adequate linguistic and world knowledge are substantial.

Current domain-independent approaches to extracting semantic information from text make heavy use of annotated corpora. These approaches require that an annotation scheme be designed, debugged, and tested against human annotators provided with an annotation environment, with inter-annotator reliability being used as a yardstick for whether the annotation task and guidelines are well-defined and feasible for humans to execute. A mixed-initiative approach that combines machine and human annotation can then be used to annotate a corpus, which is in turn used to train and test statistical classifiers to carry out the annotation task.

The above corpus-driven methodology is expensive in terms of engineering cost. There are various ways of lessening the expense, including trading off quality for quantity. For example, a system can be trained from a very large sample of fully automatic annotations and a smaller sample of human-validated annotations. Nevertheless, the total cost of putting together an annotation scheme and applying it to produce a high-quality annotated corpus is still high.

Temporal information extraction offers an interesting case study. Temporal information extraction is valuable in question-answering (e.g., answering 'when'

questions by temporally anchoring events), information extraction (e.g., normalizing information for database entry), summarization (temporally ordering information), etc. Here, as we shall see, a system has to strive for a relatively deep representation of meaning. However, the methodology outlined above breaks down to some extent when applied to this problem. This in turn suggests new approaches to annotation by humans and machines.

2 Temporal information extraction

To illustrate the problem of Temporal Information Extraction, consider the following discourse:

(1) Yesterday, John fell. He broke his leg.

A natural language system should be able to **anchor** the falling event to a particular time (yesterday), as well as **order** the events with respect to each other (the falling was *before* the breaking). We can see here that a system needs to be able to interpret events (or more generally, events and states,together called eventualities), tense information, and time expressions. The latter will be lumped under temporal adverbials,including temporal prepositions, conjunctions, etc. Further, in order to link events to times, commonsense knowledge is necessary. In particular, we infer that the breaking occurred the same day as the falling, as a result of it, and as soon as the fall occurred. However, this is a default inference; additional background knowledge or discourse information might lead to an alternative conclusion.

Consider a second example discourse (2):

(2) Yesterday Holly was running a marathon when she twisted her ankle. David had pushed her.

Here we need to understand that the use of the progressive form (i.e., aspectual information) indicates that the twisting occurred *during* the 'state' of running the marathon. Knowledge of tense (past perfect) suggests that the pushing occurs *before* the twisting (at least). Commonsense knowledge also suggests that the pushing occurs *before* and caused the twisting. We can see that even for interpreting such relatively simple discourses, a system might require a variety of sources of linguistic knowledge, including knowledge of tense, aspect, temporal adverbials, discourse relations, as well as background knowledge. Of course, other inferences are clearly possible, e.g., that the running stopped *after* the twisting, but when viewed as defaults, these latter inferences seem to be more easily violated.

Consider now the problem of representing the structure of the extracted information. It is natural to think of this in terms of a graph. For example, a graph for (1) is shown in Figure 1; here we assume the document publication date is 18 February 2004:

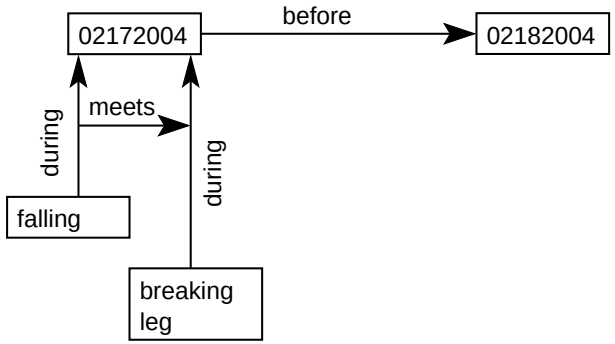

Figure 1: *Graph for a simple story*

Here we have assumed that the falling culminates in the breaking of the leg, i.e., that there is no time gap in between.

Turning to the structure of such graphs, it should be clear that the events will not necessarily be totally ordered in time, so we should consider the events and times in the graph to be partially ordered. Let us assume that events and times are represented as intervalsmarked by pairs of time points, and let us adopt the thirteen relations that Allen (1984) proposes in his interval-based temporal logic. Then, we can consider how to map NL texts to such graphs by an automatic procedure, and then use the graphs to answer questions, produce summaries, timelines, etc. The focus in this paper is on the mapping, rather than the use.

3 Previous research

Until recently, most of the prior research on temporal information extraction had drawn inspiration from work in linguistics and philosophy, as well as research on temporal reasoning in artificial intelligence. The early work of Moens & Steedman (1988) and Passonneau (1988) focused on linguistic models of event structure and tense analysis to arrive at temporal representations. For example, in Moens & Steedman (1988), "Harry hiccupped for three hours" would be analyzed as a process of iteration of the point event of hiccupping. Passonneau (1988) developed an information extraction system that could temporally locate events in texts, processing sentences like "The compressor failed before the pump seized". Much of the early work also adopted Allen's temporal relations, and used meaning representations augmented with temporal variables (Reichenbach 1947) or temporal operators (Prior 1968).

Earlier work also devoted a lot of attention to temporal aspects of discourse. A default assumption that runs through the literature (see, especially Dowty

(1986)) is that a simple past tense sentence, if it describes an event, advances the narrative, so that the event occurs after the eventuality in the previous sentence. This is the *narrative convention*of narrating events in the order they occur. If the eventuality is a state, a default assumption is that it *overlaps* with the eventuality of the previous sentence. Work by Webber (1988) related the ordering principles to a general model of discourse processing where tense was treated anaphorically, specifying a number of rules governing the temporal relationships among successive clause pairs in a discourse. Later work by Song & Cohen (1991) extended Webber's work in an implemented system that hypothesized that only certain kinds of tense shifts were coherent. They went on to suggest various heuristics to resolve ambiguities in temporal ordering. Hwang & Schubert (1992) implemented a system based on a framework of compositional semantics, showing why compositionality was a crucial property in temporal interpretation, especially for handling subordinated events.

In parallel, developments in formal semantics led to the evolution of Discourse Representation Theory (Kamp & Reyle 1993). Here the semantic representation of a sentence in a discourse context includes temporal ordering and inclusion relations over temporal indices. However, the focus was on the default narrative convention above, along with the *states overlap* assumption. Clearly, discourse relations like causality, as in (2), violate this convention. This point was taken up by work by Lascarides & Asher (1993), who developed a theory of defeasible inference that relied on a vast amount of world knowledge. Hitzeman et al. (1995) argued convincingly that reasoning in this way using background knowledge was too computationally expensive. Instead, their computational approach was based on assigning weights to different ordering possibilities based on the knowledge sources involved, with semantic distance between utterances, computed based on lexical relationships, standing in for world knowledge.

The widespread use of large corpora in NLP allowed work on temporal information extraction to advance forward quite dramatically. Wiebe et al. (1998) used a corpus-based methodology to resolve time expressions in a corpus of Spanish meeting scheduling dialogs at an overall accuracy of over 80%. Other work on resolving time expressions in meeting scheduling dialogs include Alexandersson et al. (1997) and Busemann et al. (1997). In the meantime, community-wide information extraction tasks had started to show beneficial results. The MUC-7 (1998) task tested accuracy in flagging time expressions, but did not require resolving their values. In flagging time expressions, however, at least 30% of the dates and times in the MUC test were fixed-format ones occurring in document headers, trailers, and copyright notices, thus simplifying the task.

Another area of work in temporal information extraction involves processing temporal questions.Androutsopoulos (2002) allowed users to pose temporal questions in natural language to an airport database, where English queries were mapped to a temporal extension of the SQL database language, via an intermedi-

ate semantic representation that combined both temporal operators and temporal indices. For example, the question "Which flight taxied to gate 4 at 5:00 pm?" would result in an interpretation where the taxiing started or ended at 5 pm. Although this effort was focused on databases, the emphasis on mapping a representation of NL meaning to a formal language that can support inference is inherent in approaches to temporal information extraction.

4 TimeML

The body of previous work suggests the need for an annotation scheme that can capture the kind of graph structure shown in Figure 1. TimeML (Pustejovsky et al. 2004) is a proposed metadata standard for markup of events and their temporal anchoring in documents that addresses this. It has been applied mainly to English news articles. The annotation scheme integrates together two annotation schemes: TIDES TIMEX2 (Ferro et al. 2000) and Sheffield STAG (Setzer & Gaizauskas 2000), as well as other emerging work (Katz & Arosio 2000). It identifies a variety of event expressions, including tensed verbs, e.g., "has left", "was captured","will resign"; stative adjectives "sunken", "stalled", "on board"; and event nominals "merger", "Military Operation", "Gulf War".

Eventualities in TimeML have various attributes, including the type of event, its tense, aspect, and other features. Temporal adverbials include *signals*, i.e., temporal prepositions ("for", "during", "on", "at", etc.) and connectives ("before", "after", "while", etc.). TimeML also represents time expressions, adding various modifications to TIMEX2, yielding an annotation scheme called TIMEX3. The main point of TimeML, however, is to link eventualities and times; for example, anchoring an event to a time, and ordering events and/or times. This is done by means of TLINK, or temporal links labeled with Allen-style temporal relations. Linking also take into account actual versus hypothetical events, e.g., (3), where the leaving is subordinated to a modal "may", and (4), where the leaving is subordinated to the saying/denying. These latter situations are addressed by means of SLINKs, or *subordinating* links. Thus, in (5) below, the saying subordinates the other events, which are in turn subordinated in the order found in the sentence.

(3) John may leave tomorrow.

(4) John said/denied that Mary left.

(5) The message to the chief of staff was meant to be taken as a suggestion that Sununi offer to resign, one highly placed source said.

Finally, TimeML also annotates *aspectual* verbs like "start (to cough)", "continue lazing about", etc. These verbs, rather than characterizing a distinct event, indicate a particular phase of another event; as a result, the aspectual verb is linked by a *aspectual* link (ALINK) to the event.

Recent work by Hobbs & Pustejovsky (2004) maps the structure of TimeML to a formal theory of time (the DAML small Time Ontology), which in turn allows formal queries to be posed to a reasoning system.

5 TIMEX2

TIMEX2 is the historically oldest segment of what is now TimeML. Although the guidelines are fairly complex, it is the relatively most robust part of the TimeML scheme. As a result, it has been applied more extensively than TIMEX3 or the rest of TimeML. It was developed originally by the DARPA TIDES program and has since been adopted by the U.S. Government in the Automatic Content Extraction (ACE) program's Relation Detection and Characterization (RDC) task, and in two ARDA TimeML summer workshops (NRRC 2004).

TIMEX2 is an annotation scheme for marking the extent of English time expressions (with TIMEX2 tags) and normalizing their values in ISO-8601 (1997) format (with a few extensions). The TIMEX2 scheme represents the meaning of time expressions expressed as time points, e.g., "yesterday" with the value *20040217*, or "the third week of October":*2000W42*. It also represents durations, e.g., "half an hour long":*PT30M*. TIMEX2 also handles fuzzy times such as "Summer of 1990":$1990SU$, where a primitive SU is used. It also distinguishes between specific and non-specific uses (the latter being a catchall for indefinite, habitual, and other cases) e.g., "*April* is usually wet":*XXXX04;non_specific*. Sets of times are represented to some extent, e.g., "every Tuesday" has a value *XXXXWXX2* with *periodicity F1W* and *granularity G1D*, where *F1W* means once a week, and *G1D* means a grain size of one day.

Annotators can be trained for TIMEX2 tagging very quickly (usually half a day of training followed by a few homework exercises). Inter-annotator accuracy,on the average, across 5 annotators annotating 193 news documents from the (TDT2 1999) corpus, is .86 F-measure in identifying time values. The F-measure for identifying tag extent (where tags start and end) is .79. The reason the value F-measures are higher than the extent F-measures is because the scorer flags occurrences of tags in a candidate annotation that occur in almost but not exactly the same position in the reference annotation as errors of extent, but nevertheless compares the values of such overlapping tags, scoring the values correct if the candidate and reference values are equivalent.

However, inter-annotator reliability on two features is low: F-measure on *granularity* is .51, and on *non-specificity* it is .25. While there were only a small sample of these latter features in the corpus (200 examples compared to 6000 examples of time values), these do indicate a problem, leading to a number of modifications, including the revised specification for sets in TIMEX3 (see below). Error analyses confirm that annotators do deviate from the guidelines and produce systematic errors, for example, annotating "several years ago" as *PXY*

(a period of unspecified years, a valid time expression) instead of *PAST_REF*; or annotating "all day" as *P1D* rather than *YYYYMMDD*.

6 TIMEX2 tagging

A variety of approaches have been developed to tag TIMEX2 expressions. I discuss one method here; others are briefly summarized later. The TIMEX2 tagger TempEx (Mani & Wilson 2000) handles both absolute times (e.g., "June 2, 2003") and relative times (e.g., "Thursday") by means of a number of tests on the local context. Lexical triggers like "today", "yesterday", and "tomorrow", when used in a specific sense, as well as words which indicate a positional offset, like "*next* month", "*last* year", "this *coming* Thursday" are resolved based on computing direction and magnitude with respect to a reference time, which is usually the document publication time. Bare day or month names ("Thursday", or "February") are resolved based on the tense of neighboring past or future tense verbs, if any. Signals such as "since" and "until" are used as well, along with information from nearby dates.

TempEx has been applied to different varieties of corpora, including broadcast news, print news, and meeting scheduling dialogs. The performance on all of these is comparable. On the 193-document TDT2 subcorpus, it obtained a .82 F-measure in identifying time values and .76 F-measure for extent.

In conjunction with work on tagging TIMEX2, word-sense disambiguation has also been carried out. For example, deciding whether an occurrence of "today" is non-specific or not can be carried out by a statistical classifier at .67 F-measure (using a Naïve Bayes classifier), which is significantly better than guessing the majority class (.58 F-measure for specific). Other types of sense temporal disambiguation have also been carried out. For example, deciding whether word tokens like "spring", "fall", etc. are used in a seasonal sense can be carried out at .91 F-measure (using decision trees), whereas just guessing seasonal all the time scores .54 F-measure.

7 TIMEX3 extensions

As mentioned earlier, the SET specification in TIMEX2 proved to be problematic for annotators. In TIMEX3, SET has been simplified to have two attributes in addition to the value: *quant* quantification over the set, and *freq* frequency within the set. Thus, we have examples like "three days every month": *P1M; quant=every; freq=P3D* and "twice a month": *P1M; freq=P2X*.

TIMEX3 also allows event-dependent time expressions like"three years after the Gulf War" to be tagged, since, unlike TIMEX2, events are tagged in TimeML. TIMEX3 in addition allows a functional style of encoding of offsets in time ex-

pressions, so that "last week" could be represented not only by the time value but also by an expression that could be evaluated to compute the value, namely, that it is the predecessor week of the week preceding the document date.

However, at the time of writing, automatic tagging of TIMEX3 has not yet been attempted, nor has inter-annotator reliability on TIMEX3 been studied, so we cannot as yet assess the feasibility of these extensions.

8 Challenges in TimeML link annotation

The annotation by humans of links in TimeML is a very challenging problem. Ordering judgments, as indicated by discourses (1) and (2) above, can be hard for a variety of reasons:

- The annotation of events other than tensed verbs. Since states are included, deciding which states to annotate can also be difficult, since the text may not state when a state stopped holding (this is an aspect of the AI *frame problem*). For example, given (6), we infer that Esmeralda was no longer hungry after the eating event, and that as far as we know nothing else changed. The guidelines call for just annotating those states which the text explicitly indicates as having changed, but specifying this is difficult.

  (6) Esmeralda was hungry. She ate three Big Macs.

- The difficulty of deciding whether a particular relation is warranted. For example, in (2) above, we recommended against committing to the twisting as *finishing* the marathon running. Determining what inference to commit to can be fairly subtle.
- The possibility of ambiguity or lack of clear indication of the relation. In such a case, the user is asked not to annotate the TLINK.
- The granularity of the temporal relations. A pilot experiment (Mani & Schiffman 2004) with 8 subjects providing event-ordering judgments on 280 clause pairs revealed that people have difficulty distinguishing whether there are gaps between events. The 8 subjects were asked to distinguish whether an event is (a) *strictly before* the other, (b) *before and extending into* the other, or (c) is *simultaneous* with it. These distinctions can be hard to make, as in the example of ordering "try on" with respect to "realize" in (7):

  (7) In an interview with Barbara Walters to be shown on ABCs "Friday nights", Shapiro said he tried on the gloves and realized they would never fit Simpson's larger hands.

  Not surprisingly, subjects had only about 72% agreement (corresponding to a low Kappa score of 0.5) on these ordering distinctions. Ignoring the

(a) versus (b) distinction raises the agreement to Kappa 0.61, which is (arguably) acceptable. This experiment shows that a coarse-grained concept of event ordering is more intuitive for humans.

- The density of the links. The number of possible links is quadratic in the number of events. Users can get fatigued very quickly, and may ignore lots of links.

To date, no inter-annotator study has been carried out on linking. However, analyses of a preliminary version of the Timebank Corpus, a collection of news documents annotated with TimeML at Brandeis University (NRRC 2004), reveal a number of interesting aspects of annotator behavior. In this corpus there were 186 documents, with 8324 eventualities and 3404 TLINKs, about 45 eventualities per document but only 18 TLINKs per document. This means that less than half the eventualities are being linked. Further, the vast majority (69%) of the TLINKs are within-sentence links. Sentences in news texts are generally long and complex, and many of these links involve an eventuality in a subordinate clause being linked to another in some other clause. Similarly, links between subordinate clauses of one sentence and a main clause of another are also found.

Overall, we expect that inter-annotator consistency is a hard-to-reach ideal as far as TLINKs are concerned. However, the following steps can improve consistency within and across annotators:

(1) Adding more annotation conventions. For example, it might be helpful to have annotation conventions for dealing with links out of subordinate clauses. Clearly, TimeML needs a certain level of training, more than would be required for TIMEX2, so adding specific conventions can make for tighter and more consistent annotation.
(2) Constraining the scope of annotation. The goal here is to restrict the number of decisions the human has to make. This could involve restricting the types of events and states to be annotated, as well as the conditions under which links should be annotated. Thus, efforts on a 'TimeML Lite' are important.
(3) Expanding the annotation using temporal reasoning. Since temporal ordering and inclusion operators like *before* and *during* are transitive and symmetric, it is possible to expand two different annotations by closure over transitive and symmetric relations, thereby increasing the possibility of overlap. This also boosts the amount of training data for link detection.
(4) Using a heavily mixed-initiative approach. Here automatic tagging and human validation go hand-in-hand, so that the annotator always starts from a pre-existing annotation that steadily improves.
(5) Providing the user with visualization tools during annotation. This can help them produce more densely connected graphs. This is borne out by results with a graphical visualization tool called TANGO (NRRC 2004)

that we have helped develop for annotation. This in turn has led to more complete annotations using temporal reasoning as above.

Figure 2 shows a sample TANGO screen.

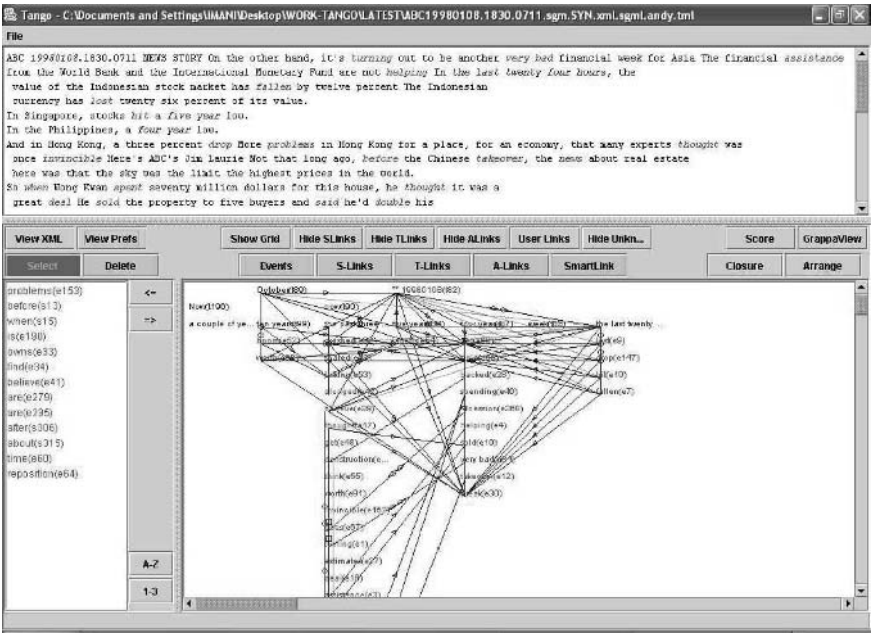

Figure 2: *TANGO: A graphical tool for annotating links*

The right-hand window shows a graphical annotation pallette, onto which events and times from the pending window on the left can be moved. The top of the pallete automatically sorts the times. The user can link events and other events or times by drawing links, with pop-up menus being used to specify non-default attributes. The system can auto-arrange the display, or rely on the user arrangement. The Closure button applies temporal reasoning rules to expand the annotation with additional links; such an expanded annotation is shown in the figure. At any point, the annotation can be dumped in XML or scored against a reference annotation.

9 Empirical constraints on temporal discourse

The availability of empirical data from experiments and corpora allow one to test to a certain extent the theories of temporal ordering discussed earlier. The tests

to date have mainly been on news. As Bell (1999) has pointed out, the temporal structure of news is dictated by perceived news value rather than chronology. Thus, the latest news is often presented first, instead of events being described in narrative order. So, one would not expect the narrative convention to be strong.

This is borne out in the experiment of Mani & Schiffman (2004) cited above, where it was found that the narrative convention applied only 47% of the time in ordering events in 131 pairs of successive past-tense clauses. Interestingly, 75% of clauses lack explicit time expressions, i.e., the 'anchor time of most events is left implicit, so that simply anchoring events to times explicitly associated with them in the text will lead to extremely poor TLINK recall. Clearly, therefore, document- and narrative-based inference could be crucial in automatic tagging.

In support of the 'states overlap principle, the TimeBank data shows that the overall percentage of links involving an *overlap* relation is 9% on the average, but 21.8% when one or both eventualities are states, a significant increase.

10 Automatic TLINK tagging

Mani et al. (2003) address the problem of implicit times by using document-level inference. Their algorithm computed a reference time (Reichenbach 1947, Kamp & Reyle 1993:594) for the event in each finite clause, defined to be either the time value of an explicit temporal expression mentioned in the clause, or, when the explicit time expression is absent, an implicit time value inferred from context, using a naive algorithm which is only 59% correct. A set of 2069 clauses from the North American News Corpus was annotated with event-time TLINK information by a human (after correcting the reference times produced by the above propagation algorithm), and then turned into feature vectors and used as training data for various machine learning algorithms. A decision rule classifier (C5.0 Rules) achieved significantly higher accuracy (.84 F-measure) compared to other algorithms as well as the majority class, where the event is simultaneous with the temporal anchor (most news events occur at the time of the explicit or implicit temporal anchor). Next, the anchoring relations and sorting of the times were used to construct a (partial) event ordering, which was evaluated by a human for document-level event orderings. The machine achieved a .75 F-measure in event ordering of TLINKs.

In comparison, Mani & Wilson (2000) used a baseline method of blindly propagating TIMEX2 time values to events based on proximity. On a small sample of 8,505 words of text, they obtained 394 correct event times in a sample of 663 verb occurrences, giving an accuracy of 59.4%. Filatova & Hovy (2000) obtained 82% accuracy on 'timestamping' clauses for a single type of event/topic on a data set of 172 clauses. However, fundamental differences between the three evaluation methods preclude a formal comparison.

While the approach of Mani et al. (2003) is corpus-based, it suffers from

several serious disadvantages, including lack of training data, very few predictive features, and rules which cover just a small number of examples. In addition, it lacks an adequate representation of discourse context in the feature vector, except for features that track shifts in tense and aspect. In future, to address this problem successfully, one would need to carry out more annotation, improve machine learning approaches, and try out a variety of other features motivated by corpus analysis.

11 Multilinguality

While the TimeML scheme in itself has been confined to English, there have been several efforts aimed at temporal information extraction for other languages. In terms of link extraction, Schilder & Habel (2000) report on a system which takes German texts and infers temporal relations from them, achieving 84% accuracy, and Li et al. (2000) take Chinese texts, and using a number of somewhat complex rules, achieves 93% accuracy on extracting temporal relations. However, these approaches are few and far between, and are hard to compare.

The problem of time expression tagging, being simpler than link extraction, has also been carried out on a number of languages. Research on time expression resolution for meeting scheduling dialogs has addressed German (Alexandersson et al. 1997, Busemann et al. 1997) as well as Spanish (Wiebe et al. 1998). The latter Spanish dialogs (from the Enthusiast Corpus of Rose et al. (1995), collected at CMU) have been translated into English and annotated with TIMEX2 tags by a bilingual annotator, based on tagging the English portion and adapting it to the Spanish.There has also been some initial work on a Hindi tagger for the TIDES Surprise Language experiment (TIMEX2 2004).

At Georgetown, we have also completed work on TIMEX2 tagging of Korean.We have annotated a corpus of 200 Korean news articles (from Hankook and Chosun newspapers) with TIMEX2. The main difference, in comparison to English TIMEX2, is in terms of morphology. Korean has agglutinative morphology, and this has implications for some of the rules for tag extent. For example, English temporal annotation guidelines state that temporal prepositions like "from" (as in "from 6 p.m.") are not part of the extent. Since Korean instead uses postpositions that are bound morphemes, we allow sub-word TIMEX2 tags that exclude the postposition. Likewise, the English guidelines require vague conjoined expressions like "three or four days", to be annotated with two tags, whereas "three or four" is a single word in Korean. Apart from this, however, the annotation scheme carries over very well. Inter-annotator reliability of 2 annotators on 30 documents shows .89 F-measure for values and extent.

Several automatic taggers have been developed at Georgetown. The first, KTX (TIMEX2 2004), is a memory-based tagger that uses a dictionary of temporal expressions and their values derived automatically from a training corpus.

Relative times in the test data are resolved using hand-created heuristics based on offset length in the training data. KTX achieves a F-measure of .66 on tagging extents and .86 F-measure for values on 200 documents. While KTX has Korean-specific morphological knowledge, it doesn't perform any prediction, being confined to just memorizing instances seen before. Another tagger, TDL (Baldwin 2001), has been developed that performs a degree of generalization. In this approach, a time expression and its TIMEX2 tag information form a training example for learning the mappings between strings and the values of temporal attribute variables. For example, a collection of similar date examples like "February 17, 2001":20010217 will generate a rule of the form $Pattern(?M, ?D, comma, ?Y) \rightarrow Value(Year(?Y), Month(?M), Day(?D))$, with a confidence based on the frequency of the pattern. TDL however lacks specific knowledge of Korean (or any other language, though it makes assumptions about the maximum word length of a time expression). TDL achieves .56 F-measure for extent and .83 F-measure for time values on 71 English documents.

Our experience with multilingual annotation suggests that the TIMEX2 scheme ports well to a variety of languages, and that a corpus-based approach with at least some language-specificity to handle morphology is, so far, the most cost-effective.

12 Conclusions

Overall, temporal information extraction offers many opportunities to tie together natural language processing and inference based on formal reasoning. The work reported here has made considerable progress due in part to the twin emphases of a corpus-based approach and evaluation. The strategy has been to develop semantic representations that can be used for formal inference in support of various practical tasks. These representations are motivated to some extent by work in formal semantics and symbolic AI. Once the representations are formally specified, the goal is then to automatically construct such representations using corpus-based methods. A similar strategy can be taken to advance the field of spatial information extraction.

However, it should be borne in mind that annotating data with relatively more complex representations is expensive and difficult to carry out. As a result, the emphasis shifts towards tools to help the human efficiently produce annotated corpora. Some of these corpora and tools are available at NRRC (2004) and TIMEX2 (2004).

At Georgetown, we are continuing to push ahead with temporal information extraction (including TLINK extraction) for different languages, including Chinese. We have also developed a new approach to modeling discourse structure from a temporal point of view, on which annotation will begin in due course. Finally, we have started to apply this work to both summarization and question-answering.

Acknowledgements. We would like to thank Georgetown's Seok Bae Jang, Jennifer Baldwin and Alan Rubinstein for their work on KTX, TDL and analyses of the Timebank corpus, respectively.

REFERENCES

Alexandersson, Jan, Norbert Riethinger & Elisabeth Maier. 1997. "Insights into the Dialogue Processing of VERBMOBIL". *Proceedings of the 5th Conference on Applied Natural Language Processing* (*ANLP-97*), 33-40. Washington, D.C.

Allen, James F. 1984. "Towards a General Theory of Action and Time". *Artificial Intelligence* 23:2.123-154.

Androutsopoulos, Ion. 2002. *Exploring Time, Tense and Aspect in Natural Language Database Interfaces*. Amsterdam & Philadelphia: John Benjamins.

Baldwin, Jennifer. 2001. "Learning Temporal Annotation of French News". Master's Research Thesis. Dept. of Linguistics, Georgetown University.

Bell, Alan. 1999. "News Stories as Narratives". *The Discourse Reader* ed. by A. Jaworski & N. Coupland, 236-251. London & New York: Routledge.

Busemann, Stephan, Thierry Declerck, Abdel Kader Diagne, Luca Dini, Judith Klein & Sven Schmeier. 1997. "Natural Language Dialogue Service for Appointment Scheduling Agents". *Proceedings of the 5th Conference on Applied Natural Language Processing* (*ANLP-97*), 25-32. Washington, D.C.

Dowty, David R. 1986. "The Effects of Aspectual Class on the Temporal Structure of Discourse: Semantics or Pragmatics?". *Linguistics and Philosophy* 9.37-61.

Ferro, Lisa, Inderjeet Mani, Beth Sundheim & George Wilson. 2001. "TIDES Temporal Annotation Guidelines Draft - Version 1.02". MITRE Technical Report MTR MTR 01W000004. McLean, Virginia: The MITRE Corporation.

Filatova, Elena & Ed Hovy. 2001. "Assigning Time-Stamps to Event-Clauses". *Workshop on Temporal and Spatial Information Processing at the 39th Annual Meeting of the Association for Computational Linguistics* (*ACL'2001*), 88-95. Toulouse, France.

ISO-8601. 1997. — `ftp://ftp.qsl.net/pub/g1smd/8601v03.pdf` [Source checked in May 2004]

Hitzeman, Janet, Marc Moens & Clare Grover. 1995. "Algorithms for Analyzing the Temporal Structure of Discourse". *Proceedings of the Annual Meeting of the European Chapter of the Association for Computational Linguistics* (*EACL'95*), 253-260. Dublin, Ireland.

Hobbs, Jerry R. & James Pustejovsky. Forthcoming. "Annotating and Reasoning about Time and Events". To appear in *The Language of Time: Readings in Temporal Information Processing* ed. by Inderjeet Mani, James Pustejovsky & Robert Gaizauskas. Oxford: Oxford University Press.

Hwang, Chung Hee & Lenhart K. Schubert. 1992. "Tense Trees as the Fine Structure of Discourse". *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics* (*ACL'92*), 232-240. Newark, Delaware.

Kamp, Hans & Uwe Reyle. 1993. *From Discourse to Logic (Part 2)*. Dordrecht: Kluwer Academic.

Katz, Graham & Fabrizio Arosio. 2001. "The Annotation of Temporal Information in Natural Language Sentences". *Workshop on Temporal and Spatial Information Processing at the 39th Annual Meeting of the Association for Computational Linguistics* (*ACL'2001*), 104-111. Toulouse, France.

Lascarides, Alex & Nicholas Asher. 1993. "Temporal Relations, Discourse Structure, and Commonsense Entailment". *Linguistics and Philosophy* 16.437-494.

Li, Wenjie, Kam-Fai Wong & Chunfa Yuan. 2001. "A Model for Processing Temporal References in Chinese". *Workshop on Temporal and Spatial Information Processing at the 39th Annual Meeting of the Association for Computational Linguistics* (*ACL'2001*), 33-40. Toulouse, France.

Mani, Inderjeet & George Wilson. 2000. "Robust Temporal Processing of News". *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics* (*ACL'2000*), 69-76. Hong Kong.

Mani, Inderjeet, Barry Schiffman & Jianping Zhang. 2003. "Inferring Temporal Ordering of Events in News". *Proceedings of the Human Language Technology Conference* (*HLT-NAACL'03*), 55-57. Edmonton, Canada.

Mani, Inderjeet & Barry Schiffman. Forthcoming. "Temporally Anchoring and Ordering Events in News". To appear in *Time and Event Recognition in Natural Language* ed. by James Pustejovsky & Robert Gaizauskas. Amsterdam & Philadelphia: John Benjamins.

Moens, Marc & Mark Steedman. 1988. "Temporal Ontology and Temporal Reference". *Computational Linguistics* 14:2.15-28.

MUC-7. 1998. *Proceedings of the 7th Message Understanding Conference*. Washington, D.C.: DARPA.

NRRC. 2004. — `http://nrrc.mitre.org/` [Source checked in May 2004]

Passonneau, Rebecca J. 1988. "A Computational Model of the Semantics of Tense and Aspect". *Computational Linguistics* 14:2.44-60.

Prior, Arthur N. 1968. "Tense Logic and the Logic of Earlier and Later". *Papers on Time and Tense* ed. by A. N. Prior, 116-134. Oxford: Oxford University Press.

Pustejovsky, James, Bob Ingria, Roser Sauri, Jose Castano, Jessica Littman, Robert Gaizauskas, Andrea Setzer, Graham Katz & Inderjeet Mani. Forthcoming. "The Specification Language TimeML". To appear in *The Language of Time: Readings in Temporal Information Processing* ed. by Inderjeet Mani, James Pustejovsky & Robert Gaizauskas. Oxford: Oxford University Press.

Reichenbach, Hans. 1947. *Elements of Symbolic Logic*. London: Macmillan.

Rose, C.P., Barbara Di Eugenio, L. Levin & C. Van Ess-Dykema. 1995. "Discourse Processing of Dialogues with Multiple Threads". *Proceedings of the 33rd Annual*

Meeting of the Association for Computational Linguistics (*ACL'95*), 31-18. Cambridge, Mass.

Schilder, Frank & C. Habel. 2001. "From Temporal Expressions to Temporal Information: Semantic Tagging of News Messages". *Workshop on Temporal and Spatial Information Processing at the 39th Annual Meeting of the Association for Computational Linguistics* (*ACL'2001*), 65-72. Toulouse, France.

Setzer, Andrea & Robert Gaizauskas, 2000. "Annotating Events and Temporal Information in Newswire Texts". *Proceedings of the Second International Conference On Language Resources And Evaluation* (*LREC-2000*), 1287-1294. Athens, Greece.

Song, Fei & Robin Cohen. 1991. "Tense Interpretation in the Context of Narrative". *Proceedings of the 9th National Conference on Artificial Intelligence* (*AAAI'91*), 131-136. Anaheim, Calif.

TDT2. 1999. — `http://morph.ldc.upenn.edu/Catalog/LDC99T37.html` [Source checked in March 2004]

TIMEX2. 2004. — `http://timex2.mitre.org/` [Source checked in March 2004]

Webber, Bonnie. 1988. "Tense as Discourse Anaphor". *Computational Linguistics* 14:2.61-73.

Wiebe, Janyce M., Thomas P. O'Hara, Thorsten Ohrstrom-Sandgren & Kenneth J. McKeever. 1998. "An Empirical Approach to Temporal Reference Resolution". *Journal of Artificial Intelligence Research* 9.247-293.

Annotation-Based Finite State Processing in a Large-Scale NLP Architecture

BRANIMIR K. BOGURAEV

IBM T.J. Watson Research Center

Abstract

There are well-articulated arguments promoting the deployment of finite-state (FS) processing techninques for natural language processing (NLP) application development. This paper adopts a point of view of designing industrial strength NLP frameworks, where emerging notions include a pipelined architecture, open-ended intercomponent communication, and the adoption of linguistic annotations as fundamental analytic/descriptive device. For such frameworks, certain issues arise — operational and notational — concerning the underlying data stream over which the FS machinery operates. The paper reviews recent work on finite-state processing of annotations and highlight some essential features required from a congenial architecture for NLP aiming to be broadly applicable to, and configurable for, an open-ended set of tasks.

1 Introduction

Recent years have seen a strong trend towards evolving a notion of robust and scalable architectures for natural language processing (NLP). A quick browse through, for instance, (Cunningham 2002, Ballim & Pallotta 2002, Patrick & Cunnigham 2003, Cunnigham & Scott 2004), shows a broad spectrum of engineering issues identified within the NLP community as essential to successful development and deployment of language technologies. Without going into any detail, it is illustrative to list a representative sample of them.

With growing use of XML as data modeling language and analysis interchange transport, *uniform document modeling* addresses concerns of widely variegated natural language text sources and formats; this includes, as we will see, strategies for representing analysis results in XML too. For incremental development and reusability of function, *componentised architecture* is becoming the accepted norm. Overall system reconfigurability is facilitated by *component inter-operability*; components are managed either by integrating them within a *framework*, or by exporting functionality packaged in a *toolkit*. Broad coverage, typically requiring streamlined utilisation of background knowledge sources, is achieved by *transparent access to resources*, using common protocols.

Such principles of design, while beneficial to the overall goal of building usable systems, pose certain constraints on system-internal data structures and

object representation, and ultimately affect the algorithmic embodiment(s) of the technologies encapsulated within.

A core characteristic of present day software architectures is that they all appeal to a notion of *annotation-based representation*, used to record, and transmit, results of individual component analysis. (For consistency, and to reinforce the uniformity of annotation-based representation, we will refer to such architectural components as "annotators".) A detailed description of general-purpose annotation formats can be found in (Bird & Liberman 2001). Specific guidelines for a linguistic annotation framework have recently been formulated by Ide & Romary (Forthcoming); for a representative sample of architecture-level considerations, and motivations, for adopting annotations as the fundamental representational abstraction, see, for instance, (Grishman 1996), and more recently, (Cunningham et al. 2002, Bunt & Romary 2002, Neff et al. Forthcoming).

The case for annotations has been made, repeatedly, and any of the publications cited above will supply the basic arguments, from an architectural point of view. There are, however, additional arguments which derive from observing a larger set of contexts in which NLP architectures have been put to use. Overall, these are characterised by situations where a particular technology needs to be used well outside the environment where it was developed, possibly by non-experts, maybe even not NLP specialists. Examples of such situations can easily be found in large research laboratories, and in corporate environments where technologies developed in the laboratories are incorporated in custom 'solutions'. Ferrucci & Lally (Forthcoming) describe an architecture for unstructured information management,which is largely informed by considerations of smooth exchange of research results and emerging technologies in the NLP area. Such an architecture would be deployed, and would require to be configured, within the organisation, not only by hundreds of researchers who will need to understand, explore, and use each other's results, but by non-specialists too.[1] It is in contexts like these that questions of uniformity, perspicuity, and generality of the underlying representation format become particularly relevant.

Annotations — when adorned with an appropriate feature system (Cunningham et al. 2002), and overlayed onto a graph structure (Bird & Liberman 2001) — combine simplicity of conceptualisation of a set of linguistic analyses with representational power (largely) adequate for capturing the intricacies of such analyses. The question still remains, however, of providing a capability (within the architecture) for exploring this space of analyses. As we will argue below, finite-state (FS) matching and transduction over annotations provides such capability, together with flexibility and convenience of rapidly configuring custom

[1] A recent article in *The Economist* (June 19th, 2003), "*Is Big Blue the Next Big Thing?*", makes a related point that 'front-line' users of emerging technologies — including NLP — are information technology specialists and consultants who would be working with customers on developing solutions that incorporate these technologies.

analysis engines which can use any, and all, analyses derived by a diverse set of upstream annotators.

There are yet other considerations arising specifically from concerns of industrial-strength NLP environments including, for instance: efficiency, both of the prototyping process, and at run time; reconciling different I/O behaviour from different, yet functionally identical, components; and ability to layer progressively more complex linguistic analyses on top of simpler annotation mark-up.

Typically, and even more so recently, when efficiency is brought up as a concern, the answer more often than not is "finite-state technology". This is not surprising, as many well-articulated arguments for finite-state processing techniques focus largely on the formal properties of finite-state automata, and the gains in speed, simplicity, and often size reduction when an FS device is used to model a linguistic phenomenon (Karttunen et al. 1996, Kornai 1999). Much rests on a particularly attractive property of FS automata, namely that after combining them in a variety of ways, the result is guaranteed to be an FS automaton. The combinations are described by regular expression patterns (van Noord & Gerdemann 2000, Beesley & Karttunen 2003), and the use of such patterns in NLP is becoming the *de facto* abstraction for conceptualising (and using) finite-state subsystems. This is reinforced by the clean separation of the algorithmic aspects of FS execution (which are hidden from the user), and the purely declarative statement(s) of configurational patterns which underlie a linguistic analysis.

In the same spirit of observing that "*there is a certain mathematical beauty to finite state calculus*" (Beesley & Karttunen 2003), there is also beauty in the ease and perspicuity of rule writing (especially if rules appeal to familiar regular expression notational conventions). If a rule-based system designed on top of FS principles, were to target an annotations store, it would facilitate both the exploration of analyses space and rapid configuration of 'consumers' of the analyses inside an arbitrarily configured instance of an NLP architecture. It is this observation alone that ultimately underpins the notion of finite-state processing over arbitrary annotations streams. It is also the case, however, that manipulating annotations within a finite-state framework offers transparent solutions to many problems referred to above, like reconciling outputs from different annotators and constructing, bottom-up and incrementally, elaborate linguistic analyses.

These kinds of observations motivate the use of FS technology in a number of contemporary NLP toolkits and architectures. The following sections discuss a variety of situated frameworks which incorporate FS techniques.

2 Finite-state technology and annotations

Broadly speaking, FS technology can be embedded in an NLP architecture in one of two ways. Conventionally, a functional component inside the architecture would be built on top of a generic finite-state transduction (FST) toolkit. As an

example, consider the function of a part-of-speech (POS) tagger, a fairly common tool in most text processing pipelines. Both (Roche & Schabes 1995) and (Kempe 1997), for example, develop methods for modeling the tag disambiguation task by means of a finite-state device. Tagging, of course, is not the only function which can be realised through FS methods: see, for instance, (Pereira & Wright 1997) for FS approximations to phrase structure grammars, or (Abney 1996) and (Ait-Mokhtar & Chanod 1997) for finite state approaches to syntax.

Such encapsulation of FS technology contributes very little, architecturally, to a general purpose capability for manipulating annotations by means of FST. The fact that an FS toolkit is used to implement a particular component function does not necessarily make that toolkit naturally available to all components in the architecture. Of more interest to this discussion are the methods developed for packaging an FS toolkit as a generic annotator inside of an NLP architecture, a module which traffics in annotations and which can be configured to perform annotation matching at any point of a pipeline.

The examples above are realised, ultimately, as character-based FST systems. If these realisations were to be packaged for incorporation in an architectural pipeline, certain transformations would need to be carried out at the interfaces between the overall data model — e.g., of a token/lemma/POS internal object — and a character string encoding (later in this section we elaborate on general strategies for, and shortcomings of, such mappings).

Furthermore, while finite-state operations are defined over an (unambiguous) stream of input, conventionally characters, the set of annotations at any arbitrary point of a processing pipeline is typically organised as a lattice, with multiple ways of threading a passage through it. To the extent that an FS subsystem explicitly employs a notation for specifying FS rules over annotations, such as for instance that in (Wunsch 2003), the question of picking a particular route through the lattice is not addressed at all (we will return to this in Section 3 below).

Cunningham et al. (2000) claim that notwithstanding the non-determinism arising from such a traversal, "*...this is not the bad news that it seems to be....*" Maybe so, but the reason is due to the fact that many grammars targeting certain annotations configurations tend to be written to exploit a carefully designed system of annotations which model relatively flat analyses. The statement will not necessarily hold for grammars written to explore an annotations store, without prior knowledge of which annotator deposited what in there. Also, in 'tightly designed' systems annotations tend to be added, monotonically, on top of existing annotations, with longer spans neatly covering shorter ones. When such a configurational property holds, it is possible to design a grammar in a way which will match against all annotations in the (implied) tree. This is, in fact, a crucial assumption in systems like Abney's (1996) CASS and SRI's FASTUS (Hobbs et al. 1997).

Such an assumption, however, will not necessarily be true of systems where a diverse set of annotators may populate the annotations store with partially overlapping and/or non-contiguous annotations. Note that a number of componentised, distributed architectures used in diverse application environments (as the ones discussed in Section 1) fall in that category.

Still, it is not uncommon to use a character-based FS toolkit to explore some annotation spaces. In essence, the basic idea is first to flatten an annotation sequence into a character string, 'exporting' relevant properties of the annotations into the string image, and then run suitably configured transducers over that string. Matching over an annotation of a certain type would be by means of defining a 'macro', which is sensitive to the string characterisation of the relevant annotation type (label) and property (or properties): thus, for example, test for a POS tag of a token annotation could be realised as a regular expression skipping over what are known to be the token characters and focusing instead on the tag characters, identified by, say, some orthographic convention (see below). A successful match, when appropriate, would insert a pair of begin-end markers bracketing the span, possibly adorned with a label to indicate a newly created type. New types can be enriched with features, with values computed during the composition of the subsumed types. Higher-level abstractions can be created then by writing rules sensitive to such phrase markers and labels. After a sequence of transducers has run, the resulting string can be 'parsed' back into the annotations store, to absorb the new annotations and their properties from the markers, labels, and feature-values.

Without going into details, the following illustrates (assuming some relatively straightforward regular grammars defining noun and verb group contours) the result of a transduction which maps a string derived from part-of-speech annotations over text tokens into a string with demarcated syntactic noun and verb groups, with instantiated morphosyntactic properties.

```
"The/DT Russian/JJ-C-S executive/JJ-C-S branch/NN sees/VB+3S an/DT
 opportunity/NN to/TO show/VB+I Russia/NP 's/POS relevance/NN"

"[NG:sng The/DT Russian/JJ-C-S executive/JJ-C-S branch/NN NG]
 [VG:+3S sees/VB+3S VG] [NG:sng an/DT opportunity/NN NG]
 [VG:inf to/TO show/VB+I VG]
 [NG:sng:poss Russia/NP 's/POS relevance/NN NG]"
```

This kind of process builds upon ideas developed in (Grefenstette 1999)[2] and (Ait-Mokhtar & Chanod 1997), with the notion of special 'markers' to constrain the area of match originally due to Kaplan & Kay (1994). Boguraev (2000) develops the strategy for using the transduction capabilities of a character-based FS toolkit (INTEX; Silberztein 1999) inside of a general-purpose NLP architecture with an abstract data model, in mediating between an annotations store and a character string.

[2] Grefenstette's term for this is "*light parsing as finite-state filtering*."

A number of problems arise with this approach. Some affect adversely the performance (which is an issue for successful deployment): character input/output is massively inefficient, and (logical) match against a high-level constituent — consider, in the above example, rules to match an `NG` or a `VG` — requires scanning over long character sequences, with cumulative expense costs. Other problems arise from cumbersome overheads: decisions need to be made concerning the (sub-)set of features to export; extra care has to be taken in choosing special characters for markers[3] which will be guaranteed not to match over the input; custom code has to be written for parsing the output string and importing the results into the annotations store, being especially sensitive to changes to its initial (pre-transduction) state; no modifications should be allowed to the character buffer underlying the input text, as chances are that *all* annotations in the system are defined with respect to the original text.

Fundamentally, however, this approach breaks down in situations where the annotations are organised in a lattice, as it requires committing to a particular path in advance: once an annotation sequence has been mapped to a string, traversal options disappear. Also, to the extent that ambiguity in the input is to be expected, a mapping strategy like the one outlined above can handle category ambiguity over the same input segment (such as part-of-speech ambiguities over tokens), but not different partitioning of the input, with segments of different length competing for a 'match' test at any point in the lattice traversal.

3 Pattern matching over annotations

The strategy for adapting a character-based FS framework to an annotations store outlined in the previous section offers a way of 'retro-fitting' FS operations into an annotations-based architecture. As we have seen, this does not meet the goal of having an infrastructure which, by design, would treat structured annotations as input 'symbols' to a finite-state calculus, and would be programmable to control the non-determinism arising from the lattice-like nature of an arbitrary annotations store.

A number of recent architectural designs incorporate a pattern matching component directly over annotations. Typically, this operates as a rule-based system, with rules' left-hand-side (and, possibly, right-hand-side too; see 3.3 below) specifying regular expressions over annotation sequences. The recognition power of such systems is, therefore, no greater than regular.

Annotations are assumed to have internal structure, defined as clusters of property-value pairs. (As we will see later, where special purpose notations are developed to define annotation 'symbols' to a grammar rule, annotation struc-

[3] In contrast, consider Kaplan & Kay's `"<"` and `">"`, which are truly special, as they are external to the alphabet.

tures tend to be flat, without embedding of lower level annotations as values to properties of higher level ones.) Both annotations and properties can be manipulated from the rules. Appealing to the formal notion of composition of FST's, patterns (or pattern grammars) can be phased into cascades. The underlying interpretation engine can be instructed to operate in a variety of matching/control regimes.

3.1 *Finite-state cascades: Background*

Organising grammars in sequences (cascades) for finite-state parsing is largely associated with the work of Hobbs et al. (1997) and Abney (1996), but finite-state methods were applied in this domain as early as 1958 (Joshi & Hopely). Abney's CASS develops a notation for regular expressions directly over syntactic categories, specifically for the purpose of partial parsing by finite-state cascades; his rewrite rules are thus defined in syntactic terms, and there is only a conceptual mapping between a grammar category and a linguistic annotation. CASS is not an architecture for NLP, and it would be hard to argue that the notation would generalise. The FASTUS system, on the other hand, as developed by Hobbs et al. is closer to the focus of this paper: the TIPSTER framework motivated one of the earlier definitions of annotations-based substrate of an NLP architecture (Grishman 1996), and within that framework, the insights gained from developing and configuring FASTUS were incorporated in the design of a Common Pattern Specification Language (CPSL; Cowie & Appelt 1998).

CPSL evolved as a pattern language over annotations, and does not fully map onto a functionally complete finite-state toolkit. In particular, there is the question whether the formalism is declarative: the language allows for function invocation *while* matching. Also, the provisions for specifying context as pre- and post-fix constraints, coupled with the way in which rulesets are associated with grammar 'phases' (a ruleset for phase is considered as a single disjunctive operation), suggests that there is no compilation of a single automaton for the entire ruleset. Furthermore, there is no notion of *transduction*, as an intrinsic FS operation; instead, similar effect is achieved by binding matched annotations, on the left hand side (LHS), and using the bound annotations on the right hand side (RHS).[4] However, the decoupling of binding from subsequent use, and the non-declarative nature of rules, additionally compounded by some syntactic ambiguity in the notation, introduces semantic problems for caching annotations.

Altogether, CPSL is tailored to the task of matching over linear sequences of annotations (as opposed to exploring tree-shaped annotation structures) and layering progressively more structured annotations over simpler ones; even so, there are limits to the complexity (or expressiveness) of the system: annotations cannot be values to other annotations' attributes.

[4] CPSL does not provide a facility for deleting annotations either.

For a while, CPSL was widely available[5], thus promoting the idea of finite-state operations over annotations with some internal structure, without having to 'bolt' such operations on top of popular character-based FST toolkits. For some interesting extensions to CPSL see, for instance, the BRIEFS project (Seitsonen 2001): where regular expression matching over tokens is added to the language (useful if the architecture does not support tokenisation), wild carding over category/ annotation attributes is supported, and most importantly, the need for matching over tree shapes is motivated, and defined (even if in somewhat rudimentary form) as an operation within the notation.

3.2 *Matching over annotations mark-up*

A recent trend in NLP data creation and exchange — the use of explicit XML markup to annotate text documents — offers a novel perspective on structured annotation matching by finite-state methods. XML naturally traffics in tree structures, which can be viewed as explicit representations of text annotations layered on top of each other. To the extent that a system can be assumed not to require partially overlapping, or stand-off, annotations, it is possible to make use of XML (with its requisite supporting technology, including e.g., schemas, parsers, transformations, and so forth) to emulate most of the functions of an annotations store. In such an approach, annotation properties are encoded via attributes; the tree configuration also supports an encoding whereby annotations can be, in effect, viewed as properties of other (higher level) annotations. It is not too hard then to conceive of a combination of some finite-state machinery, tailored to annotations, with a mechanism for traversing the XML tree as enabling technologies for flexible matching over annotation sequences — and, indeed, tree shapes — in an annotations store.

This is the insight underlying the work of Grover et al. (2000) and Simov et al. (2002). In essence, the notion is to develop a framework for defining and applying cascades of regular transducer grammars over XML documents. The components of such a framework are an XML parser, a regular grammar interpreter, a mechanism for cascading grammars, a set of conventions of how to define and interpret an input stream for a particular grammar in a cascade, and a way of traversing the document tree so as to identify and focus on the next item in that stream.

Grover et al.'s approach emphasises the utility of a general purpose transducer, `fsgmatch`, for analysis and transformations over XML documents. `fsgmatch` uses a grammar notation defined to operate over XML elements, viewing them both as atomic objects (manipulated as annotation configurations) and as strings (as targets to regular expression matches). The notation further incorporates primitives for identifying elements of interest, at a specific level and posi-

[5] Courtesy of Doug Appelt: `http://www.ai.sri.com/~appelt/TextPro/`

tion in the XML tree, and for specifying how a new XML markup element — in effect, a new annotation — is to be created from the components of a successful match. `fsgmatch` is embedded, with other tools, inside of a toolkit for encoding and manipulating annotations as XML markup over documents; the tools operate over XML files, and are combined in a pipeline. One particular rendering of such a pipeline can be viewed as a cascade of regular grammars. A core part of the toolkit is a query language which mediates, external to any tool, the target element/sub-tree which constrains the grammar application.

Even if somewhat rudimentary in this particular approach, the introduction of such a query language in an annotations-matching framework marks an important extension of the notion of FS processing over annotations: in addition to allowing querying of left/right context for a rule (via a mechanism of 'constraints'), rules can also be made sensitive to (some of the) 'upper' context of the element of interest (by targeting a grammar to a subset of all possible sub-trees in the XML document).

Simov et al.'s CLaRK system facilitates corpus and processing, similarly starting from the position that linguistic annotations are to be represented by means of XML markup. Like `fsgmatch`, the regular grammars in CLaRK operate over XML elements and tokens; unlike `fsgmatch`, CLaRK offers a tighter integration of its FS operations. This is manifested in the uniform management of hierarchies of token (and element) types, and in the enhancement of the underlying finite-state engine for regular grammar application and interpretation, with native facilities for composing grammars in a cascade and for navigating to the element(s) of interest to any particular grammar (rule). CLaRK's insight is to use XPath — a language for finding elements in an XML tree — as the mechanism both for specifying an annotation of interest to the current match, and for mapping that into an arbitrarily detailed projection of that annotation and its properties: a projection which can take the form of one or more character string sequences.

Still, this kind of mapping is established outside of the grammar; in order to specify a rule correctly, one needs to have detailed grasp of XPath to be able to appreciate the exact shape of the element and value stream submitted to the grammar. Furthermore, since the input stream to a rule is now a sequence of strings, a grammar rule is far from perspicuous in conceptualising the precise nature of the match over the underlying annotation.

And that, fundamentally, is the problem with matching over annotations represented by XML: at its core, the operation is over strings, and similarly to the approach described in Section 2, it requires making explicit, in a string form, a particular configuration and contents of annotations arising in a document processing pipeline at any moment of time. In addition to opaque notational conventions, this calls for repeated XML parsing, which may turn out to be prohibitively expensive in large scale, production-level NLP architectures. And even if the

XML-related processing overheads were to be put aside, the fact that only non-overlapping, strictly hierarchical, and in-line annotations can be rendered for matching is limiting in itself.

3.3 *Matching over structured annotations*

A different class of systems, therefore, explicitly address the issues of overlaying finite-state processing technology on top of structured annotations which are 'first-class citzens' in their respective architecture environments. The best known of these is GATE's JAPE (Java Annotation Patterns Engine; Cunningham et al. 2000). Derivative of CPSL, JAPE nonetheless stands in a class of its own: primarily because the architecture it is embedded in promotes — in a systematic and principled way — the notion of annotations as structured objects (Cunningham et al. 2002). Like CPSL, JAPE has notions like matching over linear sequences of annotations, where annotations are largely isomorphic to a broadly accepted format[6] (Bird et al. 2000); manipulating annotations by means of named bindings on the LHS of rules; and querying of left/right context. Beyond CPSL's core set of facilities, JAPE tightens up some ambiguitites in the earlier notation specification, provides utilities for grammar partitioning and rule prioritising, allows for specification of different matching regimes, and encourages factoring of patterns by means of macros.

JAPE acknowledges the inherent conflict between the descriptive power of regular expressions and the larger complexity of an annotations lattice; this is characteristically reconciled by separating RHS activation from LHS application,[7] and by assuming that components 'upstream' of JAPE will have deposited annotations so that the lattice would behave like a flat sequence. As already discussed (Section 1), such an assumption would not necessarily hold in large-scale architectures where arbitrary number of annotators may deposit conflicting/partially overlapping spans in the annotations store. To a large extent, JAPE's infrastructure (e.g., the mechanisms for setting priorities on rules, and defining different matching regimes) is intended to minimise this problem. Additionally, the ability to execute arbitrary (Java) code on the RHS is intended to provide flexible and direct access to annotations structure: operations like attribute percolation among annotations, alternative actions conditioned upon feature values, and deletion of 'scratch' annotations are cited in support of this feature. These are clearly necessary operations, but there are strong arguments to be made (see, for instance, the discussion of CPSL earlier in this section, 3.2) against dispens-

[6] Annotation components include a type, a pair of pointers to positions inside of the document content, and a set of attribute-value pairs, encoding linguistic information. In GATE, attributes can be strings, values can be any Java object.

[7] While maintaining recognition power to not beyond regular, this characterises JAPE as a pattern-action engine, rather than finite-state transduction technology.

ing with the declarative nature of a formalism, especially if the formalism can be naturally extended to accomodate them 'natively' (see Section 4 below).

A particularly effective mix of finite-state technology and complex annotation representations is illustrated by DFKI's SPPC system (Shallow Processing Production Center; Neumann & Piskorski 2000, 2002). It treats annotations with arbitrarily deep structures as input 'atoms' to a finite-state toolkit, derived from a generalisation of weighted finite-state automata (WFSA) and transducers (WFST; Mohri 1997). The toolkit functionality (Piskorski 2002) has been expanded to include operations of particular relevance to the realisation of certain text processing functions as finite-state machines, such as, for instance, Roche & Schabes' (1995) algorithm for local extension, essential for Brill-style deterministic finite-state tagging.

The breadth of the toolkit allows principled implementation of component functions in a text processing chain: SPPC's tokeniser, for instance, defers to a token classifier realised as a single WFSA, itself derived from the union of WFSA's for many token types. By appealing directly to library functions exporting the toolkit functionality, the tokeniser produces a list of token objects: triples encapsulating token start/end positions and its type.

SPPC encapsulates acascade of FSA's/FST's, whose goal is to construct a hierarchical structure over the words in a sentence. The FS cascade naturally maps onto levels of linguistic processing: tokenisation, lexical lookup, part-of-speech-tagging, named entity extraction, phrasal analysis, clause analysis. The automata expect a list of annotation-like objects — such as the token triples — as input, and are defined to produce a list of annotation-like objects as output. The granularity and detail of representation at different levels are, however, different: the lexical processor, for instance, overlays a list of lexical items on top of the token list, where a lexical item (annotation) is a tuple with references to its initial and final token objects.

Thus, each cascade level is configured for scanning different lists, and employs predicates (on its automata transition arcs) specific to each level. The hierarchical structure constructed over the sentence words is maintained explicitly by means of references and pointers among the different level lists. In fact, Neumann & Piskorki (2002) describe SPPC as operating over a complex data structure called *text chart*, and only upon abstracting away from SPPC's details it is possible to imagine this structure being conceptually equivalent to an annotations store. It is important to realise, however, that SPPC is not a text processing architecture, with uniform concept of annotations and annotation abstractions specified on arcs of FS automata. Instead, SPPC is an application, configured for a particular sequence of text processing components with a specific goal in mind, and whose custom data structure(s) are exposed to an FST toolkit by appropriately interfacing to its library functions. In other words, while SPPC demonstrates the elegance inherent to manipulating annotations by means of FS operations,

at the same time it falls short of encapsulating this in a framework which, by appealing to an annotations-centric notation would enable the configuration of arbitrary cascades over arbitrary annotation types.

Similar shortcoming can be observed of a recent system, also custom-tailored for a class of information extraction applications (Srihari et al. 2003). Like SPPC, InfoXtract maintains its own custom data structure, a *token list*, which in reality incorporates a sequence of tree structures over which a graph is overlaid for the purposes of relational information. There is clearly an interpretation of the token list as an annotations store (ignoring, for the moment, the relation encoding). For the purposes of traversing this data structure, a special formalism is developed, which mixes regular with boolean expressions. The notion is not only to be able to specify 'transductions' over token list elements, but also to have finer control over how to traverse a sequence of tree elements. This is what sets InfoXtract apart: grammars in its formalism are compiled to *tree walking automata*, with the notation providing, in addition to test/match instructions, direction-taking instructions as well.

Certain observations apply to the approaches discussed in this section. Not all configurations of annotations can be represented as hierachical arrangements: consequently, the range of annotation sequences that can be submitted to an FS device is limited. Attempts for more complex abstractions typically require ad-hoc code to meet the needs for manipulating annotations and their properties; even so, there are certain limitations to the depth of property specification on annotations. This leads to somewhat simplistic encapsulating of an 'annotation' abstraction, which even if capable of carrying the representational load of a tightly coupled system, may be inadequate for supporting co-existing — and possibly conflicting — annotations-based analyses from a multitude of 'third party' annotators. While most systems cater for left/right context inspection, examination of higher and/or lower context is, typically, not possible. In general, there is no notion of support for directing a scanner in choosing a path through the annotations lattice (apart from, perhaps, extra-grammatical means for choosing among alternative matching regimes).

4 A design for annotations-based FS matching

A particular design described here seeks to borrow from a combination of approaches outlined in the previous section; the design is, however, driven by the requirements of robust, scalable NLP architecture (as presented in Section1), and described in (Neff et al. Forthcoming). Briefly, the TALENT system (Text Analysis and Language ENgineering Tools) is a componentised architecture, with processing modules (annotators) organised in a (reconfigurable) pipeline. Annotators communicate with each other only indirectly, by means of annotations posted to, and read from, an annotation repository. In order to strictly maintain

this controlled mode of interaction between annotators, the document character buffer logically 'disappears' from an annotator's point of view. This strongly mandates that FS operations be defined over annotations and their properties.

Annotations encapsulate, in effect, data modeled as a typed feature system.Types are partitioned into families, broadly corresponding to different (logical) levels of processing: tag (e.g., HTML or XML) markup, document structure, lexical (tokens or multi-words), semantic (e.g., ontological categories), syntactic, and discourse. Features vary according to type, and the system supports dynamic creation of new types and features.

Any configuration of annotations over a particular text fragment is allowed, apart from having multiple annotations of the same type over the same span of text. Annotations of different types can be co-terminous, thanks to a priority system which makes nesting explicit, and thus facilitates encoding of tree structures (where appropriate).

A uniform mechanism for traversing the annotation repository is provided by means of custom iterators with a broad set of methods for moving forward and backward, from any given position in the text, with respect to a range of ordering functions over the annotations (in particular, start or end location, and priority). In addition, iterators can be 'filtered' by family, type, and location. As a result, it is possible to specify, programmatically, precisely how the annotations lattice should be traversed. This underlies one of the distinguishing features of our FST design: rather than rely exclusively (as, for instance, JAPE does) on specifying different control regimes in order to pick alternative paths through a lattice, we provide, inside of the FS formalism, notational conventions for directing the underlying scan (which exploit the iterators just described). This is similar to INFOXTRACT's notion of directive control, but broadly defined in terms of annotation configurations, uncommited to e.g., a tree structure.

Finite-state matching, as a system-level capability, is provided by packaging FS operations within a (meta-)annotator: TALENT's FS transducer (henceforth TFST) encapsulates matching and transduction capabilities and makes these available for independent development of grammar-based linguistic filters and processors. TFST is configurable entirely by means of external grammars, and naturally allows grammar composition into sequential cascades. Unlike other annotators, it may be invoked more than once. Communication between different grammars within a cascade, as well as with the rest of the system (including possible subsequent TFST invocations) is entirely by means of annotations; thus, in comparison to mapping annotations to strings (cf. Sections 2 and 3 above), there are no limitations to what annotation configurations could be inspected by an FS processor.

An industrial strength NLP architecture needs to identify separate components of its overall data model: in TALENT, annotations are complemented by e.g., a lexical cache, shared resources, ontological system of semantic categories,

and so forth; see (Neff et al. Forthcoming). In order for the grammar writer to have uniform access to these, the notation supports the writing of grammars with reference to all of the underlying data model.

Trying to keep up with the breadth of data types comprising a complex data model makes for increasingly cumbersome notational conventions. Indeed, it is such complexities — aiming to allow for manipulating annotations and their properties — that can be observed at the root of the design decisions of systems discussed in Section 3. The challenge is to provide for all of that, without e.g., allowing for code fragments on the right-hand side of the rules (as GATE does), or appealing to 'back-door' library functions from an FST toolkit (as SPPC allows). Both problems, that of assuming that grammar writers would be familiar with the complete type system employed by all 'upstream' (and possibly third party) annotators, and that of exposing to them API's to an annotations store have already been discussed already.

Consequently, we make use of an abstraction layer between an annotation representation (as it is implemented, in the annotation repository) and a set of annotation property specifications which relate individual annotator capabilities to granularity of analysis. Matching against an annotation — within any family, and of any particular type — possibly further constrained by attributes specific to that type, becomes an atomic transition within a finite state device. We have developed a notation for FS operations, which appeals to the system-wide set of annotation families, with their property attributes. At any point in the annotations lattice, posted by annotators prior to the TFST plugin, the symbol for current match specifies the annotation type (with its full complement of attributes) to be picked from the lattice and considered by the match operator. Run-time behaviour of this operator is determined by a symbol compiler which uses the type system and the complete range of annotation iterators (as described above) to construct, dynamically (see below), the sequence of annotations defined by the current grammar as a particular traversal of the annotations lattice, and to apply to each annotation in that sequence the appropriate (also dynamically defined) set of tests for the specified configuration of annotation attributes.

Within the notation, it is also possible to express 'transduction' operations over annotations — such as create new ones, remove existing ones, modify and/or add properties, and so forth — as primitive operations. By defining such primitives, by conditioning them upon variable setting and testing, and by allowing annotations for successful matches to be bound to variables, arbitrary manipulation of features and values (including feature percolation and embedded references to annotation types as feature values) are made possible. (This, in its own right, completely removes the need for e.g., allowing code fragments on the RHS of grammar rules.) Full details concerning notation specifics can be found in (Boguraev & Neff 2003).

The uniform way of specifying annotation types on transitions of an FST

graph hides from the grammar writer the system-wide design features separating the annotation repository from other components of the data model. For instance, access to lexical resources with morpho-syntactic information, or to external repositories like lexical databases or gazetteers, appears to the grammar writer as querying an annotation with morpho-syntactic properties and attribute values, or looking for an annotation defined in terms of a semantic ontology. Similarly, a rule can update an external resource by using notational devices identical to those for posting annotations.

The freedom to define, and post, new annotation types 'on the fly' places certain requirements on the FST subsystem. In particular, it is necessary to infer how new annotations and their attributes fit into an already instantiated data model. The TFST annotator therefore incorporates logic which, during initialisation, scans an FST file (generated by an FST compiler typically running in the background), and determines — by deferring to the symbol compiler — what new annotation types and attribute features need to be dynamically configured and incrementally added to the model.

As discussed earlier, the implementation of a mechanism for picking a particular path through the annotations lattice over which any given rule should be applied — an essential component of an annotation-based regime of FS matching — is made possible through the system of iterators described above. Within such a framework, it is relatively straightforward to specify grammars, for instance, some of which would inspect raw tokens, others would abstract over vocabulary items (some of which would cover multiple tokens), yet others might traffic in constituent phrasal units (with an additional constrain over phrase type) or/and document structure elements (such as section titles, sentences, and so forth).

For grammars which examine uniform annotation types, it is possible to infer, and construct (for the run-time FS interpreter), an iterator over such a type (in this example, as is the default, sentences). It is also possible to further focus the matching operations so that a grammar only inspects inside of certain 'boundary' annotations. The formalism is thus capable of fine-grained specification of higher and/or lower context, in addition to left/right context — an essential component of lattice traversal. In general, expressive and powerful FS grammars may be written which inspect, at different — or even the same — point of the analysis annotations of different types. In this case it is essential that the appropriate iterators get constructed, and composed, so that a felicitous annotation stream gets submitted to the run-time for inspection; TALENT deploys a special dual-level iterator designed expressly for this purpose.

Additional features of the TFST subsystem allow for seamless integration of character-based regular expression matching (not limited to tokens, and uniformly targeting the 'covered string' under any annotation), morpho-syntactic abstraction from the underlying lexicon representation and part-of-speech tagset (allowing for transparent change in tagsets and tagger/models), and composition

of complex attribute specification from simple feature tests (such as negation and conjunction). Overall, such features allow for the easy specification, via the grammar rules, of a variety of matching regimes which can transparently query the results of upstream annotators of which only the externally published capabilities are known.

5 Conclusion

The TALENT TFST system described in the previous section has been implemented in a framework with a fixed number of annotation families. While simplifying the task of the symbol compiler, this has complicated the design of a notation where transduction rules need to specify just the right combination of annotation family, type, and properties.

In order to be truly compliant with notions like declarative representation of linguistic information, representational transparencies with respect to different components of a data model, and ability to support arbitrary levels of granularity of a set of analyses (which might have both potentially incompatible, and/or mutually dependent, attribute sets), our framework has recently adopted a system of typed feature structure-based (TFS) annotation types (Ferrucci & Lally Forthcoming). A redesign of the formalism, to support FS calculus over TFS's, brings us close to the definition of "annotation transducers", introduced by Wunsch (2003), where matching operations are defined over feature-based annotation descriptions and the match criterion is subsumption among feature structures. This is, is itself, derivative of the SPROUT system (Drożdżyński et al. 2004). SPROUT openly adopts full-blown TFS's to replace regular expressions' atomic symbols, with the matching operation itself defined as unifiability of TFS's. Feature structure expressions are also used to specify the shape and content of the result of a transduction operation which is creating, conceptually, a new annotation type, constructed by unification with respect to a type hierarchy.

The design of TFST has benefited greatly from the research described in the first two sections of this paper. Extending this, we bring together the advantages of a flexible, principled, rich and open-ended representation of annotations with novel mechanisms for traversing an annotations lattice and deriving an annotation stream to be submitted to an FS device. On the descriptive side, this makes it possible to develop grammars capable of examining and transforming any configuration of annotations in an annotations store created under real, possibly noisy circumstances, by potentially conflicting annotators. On the engineering side, the benefits of TFS-based representations underlying non-deterministic FS automata with unification — in particular, their compilability into super-efficient execution devices — have already been demonstrated: see, for instance, Brawer 1998, who reports matching speeds of up to 21 million tokens per second.

REFERENCES

Abney, Steven. 1996. "Partial Parsing via Finite-State Cascades". *Natural Language Engineering* 2:4.337-344.

Aït-Mokhtar, Salah & Jean-Pierre Chanod. 1997. "Incremental Finite-State Parsing". *Proceedings of the Fifth Conference on Applied Natural Language Processing*, 72-79. Washington, D.C.

Ballim, Afzal & Vincenzo Pallotta, eds. 2002. *Robust Methods in Analysis of Natural Language Data* (= Special Issue of *Natural Language Engineering*, 8:2/3). Cambridge: Cambridge University Press.

Beesley, Kenneth & Lauri Karttunen. 2003. *Finite State Morphology*. Stanford, Calif.: CSLI Publications.

Bird, Steven & Mark Liberman. 2001. "A Formal Framework for Linguistic Annotation". *Speech Communication* 33:1/2.23-60.

Bird, Steven, David Day, John Garofolo, John Henderson, Christophe Laprun & Mark Liberman. 2000. "ATLAS: A Flexible and Extensible Architecture for Linguistic Annotation". *Proceedings of the Second International Conference on Language Resources and Evaluation* (*LREC-2000*), 1699-1706. Athens, Greece.

Boguraev, Branimir & Mary Neff. 2003. "The Talent 5.1 TFst System: User Documentation and Grammar Writing Manual". Technical Report RC22976, IBM T.J. Watson Research Center, Yorktown Heights, New York, U.S.A.

Boguraev, Branimir. 2000. "Towards Finite-State Analysis of Lexical Cohesion". *Proceedings of the 3rd International Conference on Finite-State Methods for NLP*, INTEX-3, Liege, Belgium.

Brawer, Sascha. 1998. *Patti: Compiling Unification-Based Finite-State Automata into Machine Instructions for a Superscalar Pipelined RISC Processor*. M.Sc. thesis, University of the Saarland, Saarbrücken, Germany.

Bunt, Harry & Laurent Romary. 2002. "Towards Multimodal Content Representation". *Proceedings of the Workshop on International Standards for Terminology and Language Resource Management at the 3rd International Conference on Language Resources and Evaluation* (*LREC-2002*) ed. by K. Lee & K. Choi, 54-60, Las Palmas, Canary Islands, Spain.

Cowie, Jim & Douglas Appelt. 1998. "Pattern Specification Language". TIPSTER Change Request — `http://www-nlpir.nist.gov/related_projects/tipster/rfcs/rfc10` [Source checked in May 2004]

Cunningham, Hamish & Donia Scott. 2004. *Software Architectures for Language Engineering* (= Special Issue of *Natural Language Engineering*, 10:4) Cambridge: Cambridge University Press.

Cunningham, Hamish. 2002. "GATE, a General Architecture for Language Engineering". *Computers and the Humanities* 36:2.223-254.

Cunningham, Hamish, Diana Maynard & Valentin Tablan. 2000. "JAPE: A Java Annotation Patterns Engine". Technical Memo CS-00-10, Institute for Language, Speech and Hearing (ILASH) and Department of Computer Science, University of Sheffield, Sheffield, U.K.

Cunningham, Hamish, Diana Maynard, Kalina Bontcheva & Valentin Tablan. 2002. "GATE: An Architecture for Development of Robust HLT Applications". *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (*ACL'02*), 168-175. Philadelphia, Pennsylvania.

Drożdżyński, Witold, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer & Feiyu Xu. 2004. "Shallow Processing with Unification and Typed Feature Structures — Foundations and Applications". *Künstliche Intelligenz* 1:17-23.

Ferrucci, David & Adam Lally. Forthcoming. "Accelerating Corporate Research in the Development, Application and Deployment of Human Language Technologies.". To appear in *Software Architectures for Language Engineering* (= Special Issue of *Natural Language Engineering*, 10:4).

Grefenstette, Gregory. 1999. "Light Parsing as Finite State Filtering". *Extended Finite State Models of Language* ed. by András Kornai (= *Studies in Natural Language Processing*), 86-94. Cambridge, U.K.: Cambridge University Press.

Grishman, Ralph. 1996. "TIPSTER Architecture Design Document". Technical report, DARPA, Version 2.2.

Grover, Claire, Colin Matheson, Andrei Mikheev & Marc Moens. 2000. "LT-TTT: A Flexible Tokenisation Tool". *Proceedings of the Second International Conference on Language Resources and Evaluation* (*LREC-2000*), 1147-1154. Athens, Greece.

Hobbs, Jerry, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel & Mabry Tyson. 1997. "FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text". *Finite-State Language Processing* ed. by Emmanuel Roche & Yves Schabes (= *Language, Speech, and Communication*), 383-406. Cambridge, Mass.: MIT Press.

Ide, Nancy & Laurent Romary. Forthcoming. "International Standard for a Linguistic Annotation Framework". To appear in *Software Architectures for Language Engineering* (= Special Issue of *Natural Language Engineering*, 10:4) Cambridge: Cambridge University Press.

Joshi, Aravind K. & Philip Hopely. 1999. "A Parser from Antiquity: An Early Application of Finite State Transducers to Natural Language Parsing". *Extended Finite State Models of Language* ed. by András Kornai (= *Studies in Natural Language Processing*), 6-15. Cambridge, U.K.: Cambridge University Press.

Kaplan, Ronald M. & Martin Kay. 1994. "Regular Models of Phonological Rule Systems". *Computational Linguistics* 20:3.331-378.

Karttunen, Lauri, Jean-Pierre Chanod, Gregory Grefenstette & Anne Schiller. 1996. "Regular Expressions for Language Engineering". *Natural Language Engineering* 4:1.305-328.

Kempe, André. 1997. "Finite State Transducers Approximating Hidden Markov Models". *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics* (*ACL-EACL'97*), 460-467. Madrid, Spain.

Kornai, Andras, ed. 1999. *Extended Finite State Models of Language*. Cambridge, U.K.: Cambridge University Press.

Mohri, Mehryar. 1997. “Finite-State Transducers in Language and Speech Processing”. *Computational Linguistics* 23:2.269-311.

Neff, Mary, Roy Byrd & Branimir Boguraev. Forthcoming. “The Talent System: TEXTRACT Architecture and Data Model”. To appear in *Software Architectures for Language Engineering* (= Special Issue of *Natural Language Engineering*, 10:4). Cambridge: Cambridge University Press.

Neumann, Günter & Jakub Piskorski. 2000. “An Intelligent Text Extraction and Navigation System”. *Proceedings of Content-Based Multimedia Information Access* (*Recherche d’Informations Assistee par Ordinateur RIAO-2000*), 239-246. Paris, France.

Neumann, Günter & Jakub Piskorski. 2002. “A Shallow Text Processing Core Engine”. *Journal of Computational Intelligence* 18:3.451-476.

Patrick, Jon & Hamish Cunningham, eds. 2003. *Workshop on The Software Engineering and Architecture of Language Technology Systems at HLT-NAACL 2003*. Edmonton, Alberta, Canada.

Pereira, Fernando & Rebecca Wright. 1997. “Finite-State Approximation Of Phrase Structure Grammars”. *Finite-State Language Processing* ed. by Emmanuel Roche & Yves Schabes (= *Language, Speech, and Communication*), 149-174. Cambridge, Mass.: MIT Press.

Piskorski, Jakub. 2002. “The DFKI Finite-State Toolkit”. Technical Report RR-02-04, German Research Center for Artificial Intelligence (DFKI), Saarbruecken, Saarland, Germany.

Roche, Emmanuel & Yves Schabes. 1995. “Deterministic Part-of-Speech Tagging with Finite-State Transducers”. *Computational Linguistics* 21:2.227-253.

Seitsonen, Lauri. 2001. “BRIEFS Information Extraction — Phase 2”. Technical report, TAI Research Center, Helsinki University of Technology, Helsinki, Finland.

Silberztein, Max. 2000. “INTEX: An Integrated FST Development Environment”. *Theoretical Computer Science* 231:1.33-46.

Simov, Kiril, Milen Kouylekov & Alexander Simov. 2002. “Cascaded Regular Grammars over XML Documents”. *Proceedings of the Second International Workshop on NLP and XML (NLPXML-2002)*, Taipei, Tawian. — `http://www.bultreebank.org/papers/XCRG-NLPXML-2002.pdf` [Source checked in May’04]

Srihari, R. K., Wei Li, Cheng Niu & Thomas Cornell. 2003. “InfoXtract: A Customizable Intermediate Level Information Extraction Engine”. *Proceedings of the Workshop on Software Engineering and Architectures of Language Technology Systems at HLT-NAACL*, 52-59. Edmonton, Alberta, Canada.

van Noord, Gertjan & Dale Gerdemann. 2001. “An Extendible Regular Expression Compiler for Finite-State Approaches in Natural Language Processing”. *Automata Implementation. 4th International Workshop on Implementing Automata, WIA ’99, Potsdam Germany, July 1999, Revised Papers* ed. by O. Boldt & H. Juergensen (= *Lecture Notes in Computer Science*, 2214), 122-141. Berlin: Springer.

Wunsch, Holger. 2003. *Annotation Grammars and Their Compilation into Annotation Transducers*. M.Sc. thesis, Univ. of Tübingen, Tübingen, Germany.

Acquiring Lexical Paraphrases from a Single Corpus

OREN GLICKMAN & IDO DAGAN

Bar Ilan University

Abstract

This paper studies the potential of extracting lexical paraphrases from a single corpus, focusing on the extraction of verb paraphrases. Most previous approaches detect individual paraphrase instances within a pair (or set) of comparable corpora, each of them containing roughly the same information, and rely on the substantial level of correspondence of such corpora. We present a novel method that successfully detects isolated paraphrase instances within a single corpus without relying on any a-priori structure and information.

1 Introduction

The importance of paraphrases has been recently receiving growing attention. Broadly speaking, paraphrases capture core aspects of variability in language, by representing (possibly partial) equivalencies between different expressions that correspond to the same meaning. Representing and tracking language variability is critical for many applications (Jacquemin 1999). For example, a question might use certain words and expressions while the answer, to be found in a corpus, might include paraphrases of the same expressions (Hermjakob et al. 2002). Another example is multi-document summarization (Barzilay et al. 1999). In this case, the system has to deduce that different expressions found in several documents express the same meaning; hence only one of them should be included in the final summary.

Recently, several works addressed the task of acquiring paraphrases (semi-) automatically from corpora. Most attempts were based on identifying corresponding sentences in parallel or 'comparable' corpora, where each corpus is known to include texts that largely correspond to texts in another corpus (see next section). The major types of comparable corpora are different translations of the same text, and multiple news sources that overlap largely in the stories that they cover. Typically, such methods first identify pairs (or sets) of larger contexts that correspond to each other, such as corresponding documents, by using clustering or similarity measuresat the document level, and by utilizing external information such as requiring that corresponding documents will be from the same date. Then, within the corresponding contexts, the algorithm detects individual pairs (or sets) of sentences that largely overlap in their content and are thus assumed to describe the same fact or event.

Lin & Pantel (2001) propose a different approach for extracting 'inference rules', which largely correspond to paraphrase patterns. Their method extracts such paraphrases from a single corpus rather than from a comparable set of corpora. It is based on vector-based similarity (Lin 1998), which compares typical contexts in a global manner rather than identifying all actual paraphrase instances that describe the same fact or event.

The goal of our research is to explore further the potential of learning paraphrases within a single corpus. Clearly, requiring a pair (or set) of comparable corpora is a disadvantage, since such corpora do not exist for all domains, and are substantially harder to assemble. On the other hand, the approach of detecting actual paraphrase instances seems to have high potential for extracting reliable paraphrase patterns. We therefore developed a method that detects concrete paraphrase instances within a single corpus. Such paraphrase instances can be found since a coherent domain corpus is likely to include repeated references to the same concrete facts or events, even though they might be found within generally different stories. The first version of our algorithm was restricted to identify lexical paraphrases of verbs, in order to study whether the approach as a whole is at all feasible. The challenge addressed by our algorithm is to identify isolated paraphrase instances that describe the *same* fact within a single corpus. Such paraphrase instances need to be distinguished from instances of *distinct* facts that are described in similar terms. These goals are achieved through a combination of statistical and linguistic filters and a probabilistically motivated paraphrase likelihood measure. We found that the algorithmic computation needed for detecting such local paraphrase instances across a single corpus should be quite different than previous methods developed for comparable corpora, which largely relied on a-priori knowledge about the correspondence between the different stories from which the paraphrase instances are extracted.

We have further compared our method to the vector-based approach of (Lin & Pantel 2001). The precision of the two methods on common verbs was comparable, but they exhibit some different behaviors. In particular, our instance-based approach seems to help assessing the reliability of candidate paraphrases, which is more difficult to assess by global similarity measures such as the measure of Lin and Pantel.

2 Background and related work

The importance of modeling semantic variability has been recently receiving growing attention. Dagan & Glickman (2004) propose a generic framework for modeling textual entailment that recognizes language variability at a shallow semantic level and relies on a knowledge base of paraphrase patterns. Consequently, acquisition of such paraphrase patterns is of great significance. Lexical resources such as WordNet are commonly utilized, however they tend to be too

broad and do not contain the necessary domain vocabulary. Similarity-based lexical approaches (Lin 1998) are also inappropriate for semantic entailment for they do not capture equivalence and entailment of meaning but rather broader meaning similarity. Several works addressed the task of automatically acquiring paraphrase patterns from corpora. (Barzilay & McKeown 2001) use sentence alignment to identify paraphrases from a corpus of multiple English translations of the same text. (Pang et al. 2003) also use a parallel corpus of Chinese-English translations to build finite state automata for paraphrase patterns, based on syntactic alignment of corresponding sentences. (Shinyama et al. 2002) learn structural paraphrase templates for Information extraction from a comparable corpus of news articles from different news sources over a common period of time. Similar news article pairs from the different news sources are identified based on document similarity. Sentence pairs are then identified based on the overlap of Named Entities in the matching sentences. (Barzilay and Lee 2003) also utilizes a comparable corpus of news articles to learn paraphrase patterns, which are represented by word lattice pairs. Patterns originating from the same day but from different newswire agencies are matched based on entity overlap. We compare our results to those of the algorithm by (Lin & Pantel 2001), which extracts paraphrase-like inference rules for question answering from a single source corpus. The underlying assumption in their work is that paths in dependency trees that connect similar syntactic arguments (slots) are close in meaning. Rather then considering a single feature vector that originates from the arguments in both slots, vector-based similarity was computed separately for each slot. The similarity of a pair of binary paths was defined as the geometric mean of the similarity values that were computed for each of the two slots.

3 Algorithm

Our proposed algorithm identifies candidates of corresponding verb paraphrases within pairs of sentences. We define a *verb instance pair* as a pair of occurrences of two distinct verbs in the corpus. A *verb type pair* is a pair of verbs detected as a candidate lexical paraphrase.

3.1 *Preprocessing and representation*

Our algorithm relies on a syntactic parser to identify the syntactic structure of the corpus sentences, and to identify verb instances. We treat the corpus uniformly as a set of distinct sentences, regardless of the document or paragraph they belong to. For each verb instance we extract the various syntactic components that are related directly to the verb in the parse tree. For each such component we extract its lemmatized head, which is possibly extended to capture a semantically specified constituent. We extended the heads with any lexical modifiers that constitute a multi-word term, noun-noun modifiers, numbers and prepositional ‘of’

subject	secretary_general_boutros_boutros_ghali	subject	iraqi_force
object	implementation_of_deal	object	kurdish_rebel
modifier	after	pp-on	august_31
(A) verb: delay		(B) verb: attack	

Figure 1: *Extracted verb instances for sentence "But U.N. Secretary-General Boutros Boutros-Ghali delayed implementation of the deal after Iraqi forces attacked Kurdish rebels on August 31."*

complements. Verb instances are represented by the vector of syntactic modifiers and their lemmatized fillers. For illustration, Figure 1 shows an example sentence and the vector representations for its two verb instances.

3.2 *Identifying candidate verb instance pairs (filtering)*

We apply various filters in order to verify that two verb instances are likely to be paraphrases describing the same event. This is an essential part of the algorithm since we do not rely on the high a-priori likelihood for finding paraphrases in matching parts of comparable corpora.

We first limit our scope to pairs of verb instances that share a common (extended) subject and object which are not pronouns. Otherwise, if either the subject or object differ between the two verbs then they are not likely to refer to the same event in a manner that allows substituting one verb with the other. Additionally, we are interested in identifying sentence pairs with a significant overall term overlap, which further increases paraphrase likelihood for the same event. This is achieved with a standard (Information Retrieval style) vector-based approach, with tf-idf term weighting

- $tf(w) = freq(w)$ in sentence
- $idf(w) = log(N/freq(w)$ in corpus $)$ where N is the total number of tokens in the corpus.

Sentence overlap is measured simply as the dot product of the two vectors. We intentionally disregard any normalization factor (such as in the cosine measure) in order to assess the absolute degree of overlap, while allowing longer sentences to include also non-matching parts that might correspond to complementary aspects of the same event. Verb instance pairs whose sentence overlap is below a specified threshold are filtered out.

An additional assumption is that events have a unique propositional representation and hence verb instances with contradicting vectors are not likely to describe the same event. We therefore filter verb instance pairs with contradicting propositional information - a common syntactic relation with different arguments. As an example, the sentence "Iraqi forces captured Kurdish rebels on August 29." Has a contradicting 'on' preposition argument with the sentence from Figure 1(B) ("August 29" vs. "August 31").

3.3 *Computing paraphrase score of verb instance pairs*

Given a verb instance pair (after filtering), we want to estimate the likelihood that the two verb instances are paraphrases of the same fact or event. We thus assign a paraphrase likelihood score for a given verb instance pair Iv_1, v_2, which corresponds to instances of the verb types v_1 and v_2 with overlapping syntactic components $p_1, p_2, \ldots, p_n$. The score corresponds (inversely) to the estimated probability that such overlap had occurred by chance in the entire corpus, capturing the view that a low overlap probability (i.e., low probability that the overlap is due to chance) correlates with paraphrase likelihood. We estimate the overlap probability by assuming independence of the verb and each of its syntactic components as follows:

$$\begin{aligned} \mathrm{P}(I_{v_1,v_2}) &= \mathrm{P}(\text{overlap}) = \mathrm{P}(v_1, p_1, \ldots, p_n)\,\mathrm{P}(v_2, p_1, \ldots, p_n) \\ &= \mathrm{P}(v_1)\,\mathrm{P}(v_2)\prod_{i=1}^{n}\mathrm{P}(p_i)^2 \end{aligned} \quad (1)$$

Where the probabilities were calculated using Maximum Likelihood estimates based on the verb and argument frequencies in the corpus.

3.4 *Computing paraphrase score for verb type pairs*

When computing the score for a verb type pair we would like to accumulate the evidence from its corresponding verb instance pairs. Following the vein of the previous section we try to estimate the joint probability that these different instance pairs occurred by chance. Assuming instance independence, we would like to multiply the overlap probabilities obtained for all instances. We have found, though, that verb instance pairs whose two verbs share the same subject and object are far from being independent (there is a higher likelihood to obtain additional instances with the same subject-object combination). To avoid complex modeling of such dependencies we picked only one verb instance pair for each subject-object combination, taking the one with lowest probability (highest score). This yields the set $T(v_1, v_2) = (I_1, \ldots, I_n)$ of best scoring (lowest probability) instances for each distinct subject and object components. Assuming independence of occurrence probability of these instances, we estimate the probability $\mathrm{P}(T(v_1, v_2)) = \prod \mathrm{P}(I_i)$, where $\mathrm{P}(I)$ is calculated by Equation (1) above. The score of a verb type pair is given by: score$(v_1, v_2) = -log\,\mathrm{P}(T(v_1, v_2))$.

4 Evaluation and analysis

4.1 *Setting*

We ran our experiments on the first 15-million word (token) subset of the Reuters Corpus. The corpus sentences were parsed using the Minipar dependency parser

(Lin 1993). 6,120 verb instance pairs passed filtering (with overlap threshold set to 100). These verb instance pairs derive 646 distinct verb type pairs, which were proposed as candidate lexical paraphrases along with their corresponding paraphrase score.

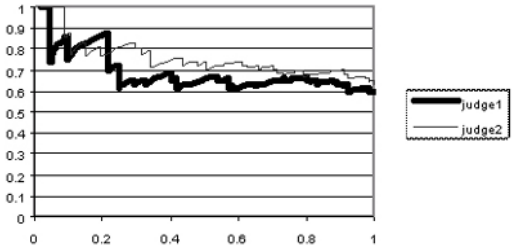

Figure 2: *Precision (y axis) recall (x axis) curves of system paraphrases by judge (verb type pairs sorted by system score)*

The correctness of the extracted verb type pairs was evaluated over a sample of 215 pairs (one third of the complete set) by two human judges, where each judge evaluated one half of the sample. In a similar vein to related work in this area, judges were instructed to evaluate a verb type pair as a *correct* paraphrase only if the following condition holds: one of the two verbs can replace the other within some sentences such that the meaning of the resulting sentence will entail the meaning of the original one. To assist the judges in assessing a given verb type pair they were presented with example sentences from the corpus that include some matching contexts for the two verbs (e.g., sentences in which both verbs have the same subject or object). Notice that the judgment criterion allows for directional paraphrases, such as ⟨invade, enter⟩ or ⟨slaughter, kill⟩, where the meaning of one verb entails the meaning of the other, but not vice versa.

4.2 *Results of the paraphrase identification algorithm*

Figure 2 shows the precision vs. recall results for each judge over the given test-sets. The evaluation was conducted separately also by the authors on the full set of 646 verb pairs, obtaining comparable results to the independ-ent evaluators. In terms of agreement, the Kappa value (measuring pair wise agreement discounting chance occurrences) between the authors and the independent evaluators' judgments were 0.61 and 0.63, which correspond to a substantial agreement level (Landis & Koch 1977). The overall precision for the complete test sample is 61.4% accuracy, with a confidence interval of [56.1,66.7] at the 0.05 significance level. Figure 3 shows the top 10 lexical paraphrases, and a sample

of the remaining ones, achieved by our system along with the annotators' judgments. Figure 4 shows correct sentence pairs describing a common event, which were identified by our system as candidate paraphrase instances.

1-	⟨fall, rise⟩	8+	⟨cut, lower⟩	302+	⟨kill, slaughter⟩
2+	⟨close, end⟩	9-	⟨rise, shed⟩	362+	⟨bring, take⟩
3+	⟨post, report⟩	10+	⟨fall, slip⟩	422+	⟨note, say⟩
4+	⟨recognize, recognize⟩	62+	⟨honor, honour⟩	482-	⟨export, load⟩
5+	⟨fire, launch⟩	122+	⟨advance, rise⟩	542+	⟨downgrade, relax⟩
6+	⟨drop, fall⟩	182+	⟨benefit, bolster⟩	602+	⟨create, establish⟩
7+	⟨regard, view⟩	242+	⟨approve, authorize⟩	632-	⟨announce, give⟩

Figure 3: *Example of system output with judgments*

An analysis of the incorrect paraphrases showed that roughly one third of the errors captured verbs with contradicting semantics or antonyms (e.g., ⟨rise, fall⟩, ⟨buy, sell⟩, ⟨capture, evacuate⟩) and another third were verbs that tend to represent correlated events with strong semantic similarity (e.g., ⟨warn, attack⟩, ⟨reject, criticize⟩). These cases are indeed quite difficult to distinguish from true paraphrases since they tend to occur in a corpus with similar overlapping syntactic components and within quite similar sentences. Figure 4 also shows examples of misleading sentence pairs demonstrating the difficulties posed by such instances. It should be noticed that our evaluation was performed at the verb type level. We have not evaluated directly the correctness of the individual paraphrase instance pairs extracted by our method (i.e., whether the two instances in a paraphrase pair indeed refer to the same fact). Such evaluation is planned for future work. Finally, a general problematic (and rarely addressed) issue in this area of research is how to evaluate the coverage or recall of the extraction method relative to a given corpus.

4.3 *Comparison with (Lin & Pantel 2001)*

We applied the algorithm of (Lin & Pantel 2001), denoted here as the LP algorithm, and computed their similarity score for each pair of verb types in the corpus. To implement the method for lexical verb paraphrases, each verb type was considered as a distinct path whose subject and object play the roles of the X and Y slots.

As it turned out, the similarity score of LP does not behave uniformly across all verbs. For example, many of the top 20 highest scoring verb pairs are quite erroneous (see Figure 5), and do not constitute lexical paraphrases (compare with the top scoring verb pairs for our system in Figure 3). The similarity scores do seem meaningful within the context of a single verb v, such that when sorting all other verbs by the LP score of their similarity to v correct paraphrases are more likely to occur in the upper part of the list. Yet, we are not aware of a criterion

correct paraphrase instance pairs

Campbell is **buying** Erasco from Grand Metropolitan Plc of Britain for about $210 million.	Campbell is **purchasing** Erasco from Grand Metropolitan for approximately US$210 million.
The stock of Kellogg Co. **dropped** Thursday after the giant cereal maker warned that its earnings for the third quarter will be 20 percent below a year ago.	The stock of Kellogg Co. **fell** Thursday after it warned about lower earnings this year ...
Ieng Sary on Wednesday formally **announced** his split with top Khmer Rouge leader Pol Pot, and said he had formed a rival group called the Democratic National United Movement.	In his Wednesday announcement Ieng Sary, who was sentenced to death in absentie for his role in the Khmer Rouge's bloody rule, **confirmed** his split with paramount leader Pol Pot.

misleading instance pairs

Last Friday, the United States **announced** punitive charges against China's 1996 textile and apparel quotas ...	China on Saturday urged the United States to **rescind** punitive charges against Beijing's 1996 textile and apparel quotas ...
Municipal bond yields **dropped** as much as 15 basis points in the week ended Thursday, erasing increases from the week before.	Municipal bond yields **jumped** as much as 15 basis points over the week ended Thursday ...
Rand Financials notably **bought** October late while Chicago Corp and locals lifted December into by stops.	Rand Financials notably **sold** October late while locals pressured December.

Figure 4: *Examples of instance pairs*

that predicts whether a certain verb has few good paraphrases, many or none. Given this behavior of the LP score we created a test sample for the LP algorithm by randomly selecting verb pairs of equivalent similarity rankings relative to the original test sample. Notice that this procedure is favorable to the LP method for it is evaluated at points (verb and rank) that where predicted by our method to correspond to a likely paraphrase.

The resulting 215 verb pairs were evaluated by the judges along with the sample for our method, while the judges did not know which system generated each pair. The overall precision on the LP method for the sample was 51.6%, with a confidence interval of [46.1,57.1] at the 0.05 significance level. The LP

⟨misread, misjudge⟩(0.62); ⟨barricade, sandbag⟩(0.29); ⟨disgust, mystify⟩(0.27); ⟨jack, decontrol⟩(0.27); ⟨Pollinate, pod⟩(0.25); ⟨mark_down, decontrol⟩(0.23); ⟨subsidize, subsidise⟩(0.22); ⟨wake_up, divine⟩(0.21); ⟨thrill, personify⟩(0.21); ⟨mark_up, decontrol⟩(0.20); ⟨flatten, steepen⟩(0.20); ⟨mainline, pip⟩(0.20); ⟨misinterpret, relive⟩(0.20); ⟨remarry, flaunt⟩(0.19); ⟨distance, dissociate⟩(0.18); ⟨trumpet, drive_home⟩(0.18); ⟨marshal, beleaguer⟩(0.17); ⟨dwell_on, feed_on⟩(0.17); ⟨scrutinize, misinterpret⟩(0.16); ⟨disable, counsel⟩(0.16)

Figure 5: *Top 20 verb pairs from similarity system*

results for this sample were thus about 10 points lower than the results for our comparable sample, but the two confidence intervals overlap slightly. It is interesting to note that the precision of the LP algorithm over all pairs of rank 1 was also 51%, demonstrating that just rank on its own is not a good basis for paraphrase likelihood. Figure 6 shows overall recall vs. precision from both

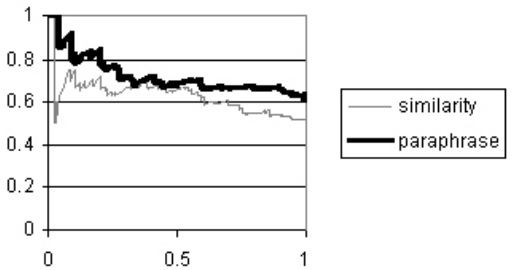

Figure 6: *Precision recall curve for our paraphrase method and LP similarity*

judges for the two systems. The results above show that the precision of the vector-based LP method may be regarded as comparable to our instance-based method, in cases where one of the two verbs was identified by our method to have a corresponding number of paraphrases. The obtained level of accuracy for these cases is substantially higher than for the top scoring pairs by LP. This suggests that our approach can be combined with the vector-based approach to obtain higher reliability for verb pairs that were extracted from actual paraphrase instances.

5 Conclusions

This work presents an algorithm for extracting lexical verb paraphrases from a single corpus. To the best of our knowledge, this is the first attempt to identify actual paraphrase instances in a single corpus and to extract paraphrase patterns directly from them. The evaluation suggests that such an approach is indeed viable, based on algorithms that are geared to overcome many of the misleading cases that are typical for a single corpus (in comparison to comparable corpora). Furthermore, a preliminary comparison suggests that verb pairs extracted by our instance-based approach are more reliable than those based on global vector similarity. As a result, an instance-based approach may be combined with a vector-based approach in order to assess better the paraphrase likelihood for many verb pairs. Future research is planned to extend the approach to handle more complex

paraphrase structures and to increase its performance by relying on additional sources of evidence.

REFERENCES

Barzilay, Regina & Lillian Lee. 2003. "Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment". *Proceedings of the Human Language Technology Conference* (*HLT-NAACL'03*), 16-23. Edmonton, Canada.

Barzilay, Regina & Kathleen McKeown. 2001. "Extracting Paraphrases from a Parallel Corpus". *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (*ACL'01*), 50-57. Toulouse, France.

Barzilay, Regina, Kathleen McKeown & Michael Elhadad. 1999. "Information Fusion in the Context of Multidocument Summarization". *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics* (*ACL'99*), 550-557. Univ. of Maryland, College Park, Maryland, U.S.A.

Dagan, Ido & Glickman Oren. 2004. "Probabilistic Textual Entailment: Generic Applied Modeling Of Language Variability". *PASCAL workshop on Text Understanding and Mining*, Grenoble, France.

Hermjakob, Ulf, Abdessamad Echihabi & Daniel Marcu. 2002. "Natural Language Based Reformulation Resource and Web Exploitation for Question Answering". *Proceedings of the Eleventh Text REtrieval Conference* (*TREC 2002*), 801-810. Gaithersburg, Maryland, U.S.A.

Jacquemin, Christian. 1999. "Syntagmatic and Paradigmatic Representations of Term Variation". *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics* (*ACL'99*), 341-348. Univ. of Maryland, College Park, Maryland, U.S.A.

Landis, J.R. & G.G. Koch. 1997. "The Measurements of Observer Agreement for Categorical Data". *Biometrics* 33:159-174.

Lin, Dekang. 1993. "Principle-Based Parsing without Overgeneration". *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics* (*ACL'93*), 112-120. Columbus, Ohio.

Lin, Dekang. 1998. "Automatic Retrieval and Clustering of Similar Words". *Proceedings of the 17th International Conference on Computational Linguistics* (*COLING/ACL'98*), 768-774. Montréal, Canada.

Lin, Dekang & Patrick Pantel. 2001. "Discovery of Inference Rules for Question Answering". *Natural Language Engineering* 7:4.343-360.

Pang, Bo, Kevin Knight & Daniel Marcu. 2003. "Syntax-based Alignment of Multiple Translations: Extracting Paraphrases and Generating New Sentences". *Proceedings of the Human Language Technology Conference* (*HLT-NAACL'03*), 181-188. Edmonton, Canada.

Shinyama, Yusuke, Satoshi Sekine, Kiyoshi Sudo & Ralph Grishman. 2002. "Automatic Paraphrase Acquisition from News Articles". *Proceedings of the Human Language Technology Conference* (*HLT'02*), 51-58. San Diego, Calif., U.S.A.

Multi-Word Collocation Extraction by Syntactic Composition of Collocation Bigrams

VIOLETA SERETAN, LUKA NERIMA & ERIC WEHRLI
Language Technology Laboratory, University of Geneva

Abstract

This paper presents a method of multi-word collocation extraction, which is based on the syntactic composition of two-word collocations previously identified in text. We describe a procedure of word linking that iteratively builds up longer expressions, which constitute multi-word collocation candidates. We then present several measures used for candidates ranking according to the collocational strength, and show the results of a trigram extraction experiment. The methodology used is particularly suited for the extraction of flexible collocations, which can undergo complex syntactical transformations such as passivization, relativization and dislocation.

1 Introduction

Collocations,defined as "arbitrary and recurrent word combinations" in (Benson 1990) or "institutionalized phrases" in (Sag et al. 2002), represent a subclass of multi-word expressions that are prevalent in language and play an important role in its naturalness. The *collocate*, i.e., the "right word" used in combination with a given word (usually called *base word*) is unpredictable for non-native speakers. The preference for a specific word instead of another is dictated by the conventional usage in a specific language, dialect, domain (or even a time period), rather than by syntactic or semantic criteria. A French speaker, for instance, needs to be aware of the conventional usage of an expression, e.g., "*encounter difficulties*", in order to avoid unnatural paraphrases such as *"*feel difficulties*". Collocations constitute a big concern for non-native speakers faced with the task of producing proficient text. In NLP, the collocational knowledge is indispensable for major applications such as the machine translation and the natural language generation.

The phenomenon of words collocability has been given particular attention since Firth (1957), who made the statement that a words is characterized by "the company it keeps". Two main approaches have been followed for the collocational knowledge acquisition: a lexicographic one, oriented towards the creation of dictionaries encoding words' combinatorial possibilities (notably (Benson et al. 1986, Mel'cuk et al. 1984)), and a statistical one, aimed at automatically extracting relevant word associations from text corpora (e.g., Choueka et al. 1983, Sinclair 1991, Church & Hanks 1990, Smadja 1993).

As stated by Harris (1988) in the "likelihood constraint" ("*each word has a particular and roughly stable likelihood of occurring as argument, or operator,*

with a given word"), the collocating words are syntactically bound. The collocation is, in fact, a well-formed expression. Unfortunately, the existing collocation extraction methods usually ignore the linguistic structure and rely almost completely on the text's surface, while it is generally agreed that the extraction should ideally be done from analyzed rather than from raw text (Smadja 1993:151).

More recent work (e.g., (Grishman & Sterling 1994, Lin 1998, Krenn & Evert 2001)) performs a shallow text analysis (POS tagging, lemmatization, syntactic dependency test) in order to syntactically filter the candidate expressions. Still, it is insufficient to account for the flexible collocations, in which the constituent words may appear inverted, arbitrarily distant from each other, and may not be directly related syntactically (as, for instance, the words "*overcome*" and "*difficulties*" in the expression "*the **difficulties** that the country tried to **overcome***").

Some major drawbacks of the classical methods are: the possible ungrammaticality of the candidates considered, the combinatorial explosion when considering all possible words combinations, the limitation (of the majority of stochastic tests) to two-word combinations. These drawbacks can only be overcome by performing a deep syntactic analysis that takes into account the complex grammatical processes underlying the text form.

The performance of extraction systems is essential for the subsequent treatment of collocations in important NLP applications such as machine translation, information retrieval, word sense disambiguation. Therefore we propose an approach to multi-word collocation extraction which focuses on the use of syntactic analysis and syntactic criteria for defining collocation candidates. Our approach is supported by the strong increase, over the last few years, of the availability of computational resources and software tools dedicated to large-scale and robust syntactic parsing[1].

The paper is organized as follows. Section 2 briefly presents some of the existing methods of multi-word collocation extraction and their main features. Section 3 outlines the method of collocation bigram extraction on which our work relies. In Section 4 we describe in detail the method we propose for extracting multi-word collocations using the collocation bigrams. In Section 5 we present the experimental results obtained by applying this method on a collection of English newspaper articles. The last section draws the conclusion and points out directions for further development.

2 Existing methods of multi-word collocation extraction

Traditional approaches to automatic collocation extraction from text corpora rely on stochastic measures,which range from the simple word co-occurrence frequency to more sophisticated statistical tests (e.g., log-likelihood ratios test

[1] See (Ballim & Pallotta 2002) for recent advances in robust parsing.

(Dunning 1993), Student's *t*-test, Pearson's χ^2 test[2]) or on information theoretic measures (e.g., the mutual information (Church & Hanks 1990)).

One feature these methodsshare is that they use the textual proximity as the main criterion for the selection of candidate collocations, instead of syntactic criteria. Since they consider any word combination as a valid candidate, they are forced to limit to a text window of fixed size (usually 5 words). Moreover, they usually do not take into account collocations made up of more than two words, as the lexical association measures are generally designed for pairs of items.

Only few methods, e.g., (Choueka et al. 1983, Smadja 1993), are also concerned with *n*-grams ($n>2$). The method proposed by Choueka et al. (1983) for finding *n*-word collocations considers the frequency of consecutive word sequences of length *n* (with *n* from 2 to 6). The limitation to n=6 is due to the rapid increase of the number of all possible *n*-grams, for *n* bigger than 6.

The Xtract system of Smadja (1993) retrieves, in a first stage, word bigrams that are not necessarily contiguous in text, but can be separated by several words. It then looks, in the second stage, at the words in the bigrams' surrounding positions and identifies *n*-grams as the repetitive contexts of already identified bigrams. These repetitive contexts can form either "rigid noun phrases", or "phrasal templates" (phrases containing empty slots standing for parts of speech).

Both methods rely only on a superficial text representation, while the authors point out that the selection of terms should ideally be done following linguistic criteria.

Since robust large-scale parsers became in the meantime available, the more recent methodsfocus on using parsed rather than raw text for bigram extraction (e.g., Lin 1998, Goldman et al. 2001). Our work relies to a large extent on the features of the method of (Goldman et al. 2001), which we will briefly present in the next section.

3 Collocation bigrams extraction with FipsCo

FipsCo (Goldman et al. 2001) is a term extractor systemthat relies on Fips (Laenzlinger & Wehrli 1991), a robust, large-scale parser based on an adaptation of Chomsky's "Principles and Parameters" theory. The system extracts, from the parsed text, all the co-occurrences of words in given syntactic configurations: noun-adjective, adjective-noun, noun-noun, noun-preposition-noun, subject-verb, verb-object, verb-preposition, verb-preposition-argument. It thus apply a strong syntactic filter on the candidate bigrams. Subsequently, it applies the log-likelihood test (Dunning 1993) on the sets of bigrams obtained, in order to rank them according to the degree of collocability.

[2] For a rather comprehensive overview see chapter 5 of (Manning & Schütze 1999).

The strength of this approach comes from the combinationof the deep syntactic analysis with the statistical test. The sentences are normalized (the words are considered in their lemmatized form and in their canonical position). The system is able to handle complex cases of extraposition, such as relativization, passivization, raising, dislocation.

To illustrate this, let us consider the following sentence fragment (a real corpus example we have encountered): "*the **difficulties** that the country tried to **overcome***". Extracting the collocation of verb-object type "*overcome — difficulty*" requires a complex syntactic analysis, made up from several steps: recognizing the presence of a relative clause; identifying the antecedent ("*the difficulties*") of the relative pronoun "*that*"; and establishing the verb-object link between this pronoun and the verb of the relative clause. This collocation will simply be overlooked by classical statistical methods, where usually the size of the collocational window is 5. Such situations are quite frequent for example in Romance languages[3], in which the words can undergo complex syntactic transformations.

4 Multi-word collocation extraction by bigrams composition

The system presented above is able to extract syntactically bound collocation bigrams, which may occur unrestrictedly with respect to the textual realization. The system relies on a deep syntactic analysis that can handle complex cases of extraposition. We will take advantage of these features for the multi-word collocations identification, since they guarantee both the grammaticality of results, and the unconstrained search space and realization form.

Since the FipsCo system actually returns not only the best scored collocations, but all the candidate bigrams, we generate all the possible multi-word associations from text. Our goal is to build up, using the set of extracted bigrams, the sequences of bigrams sharing common words. The obtained collocate chains represent well-formed multi-word associations. The configuration of their syntactic structure is defined by the syntactic relations in the bigrams involved. The shared term must be the same not only lexically, but also indexically (the very same occurrence in the text). Due to the syntactic constraint, the shared term will actually appear in the same sentence as the other collocates.

For instance, given two bigrams (w_1 w_2), (w_2 w_3) with, we can construct the trigram: (w_1 w_2 w_3), as in the case of the following collocations: "*terrorist attack*", "*attack of September*"; we obtain the trigram collocation "*terrorist attack of September*".

[3] Goldman et al. (2001) report a high percentage of cases in which the distance base-collocate is more than 5 words in a French corpus.

Note that the condition of indexical identity avoids combinations with different readings in case of polysemy, e.g., "*terrorist attack*" with "*attack of coughing*".

Repeating the same procedure we can add further words to the obtained trigrams, thus obtaining multi-words collocations of arbitrary length. Moving on to *n*-grams will conserve the inclusion of all terms in the same sentence. We impose no default restrictions on the syntactic configuration of the resulting expression.

In what follows, we present the word linking procedure that allows the construction of longer multi-word collocations (henceforth MWCs) from shorter collocations and the measures proposed for ranking them.

4.1 *Iterative word linking procedure*

The procedure of linking new words to existing collocations in order to discover longer collocations makes use of the criterion of the existence of a syntactic link between the new words and one of the existing collocation's words. Recursively applied to the set of generated collocations in each step, this procedure allows the incremental composition of longer collocation from shorter subparts. In thus leads to the identification of all collocation candidates in a text; the distance between the composing words is only limited by the sentence's boundaries.

Building up all the possible trigrams from a set of bigrams can be done, for example, by considering all the pairs of bigrams that share terms. We call "pivot" the term shared by two bigrams. There are three possibilities to construct a trigram, that correspond to the position of the pivot in the two bigrams. In the first case, the pivot is the last term in one bigram, and the first in the another. It occupies the middle (internal) position in the new trigram, as in "*terrorist **attack** of September*". In the other two cases, the pivot occupies an external position, either on the left (as in "***have** impact on*", derived from the bigrams "***have** impact*" and "***have** on*"), or in the right (as in "*round [of] presidential **election***", derived from "*round [of] **election***" and "*presidential **election***").

For the general case, the following procedure is used to incrementally build up longer *n*-grams:

$\mathcal{C} := \mathcal{D}$;
repeat
  $\mathcal{N} := \emptyset$;
  for each MWC_i **in** $\mathcal{C}$
    for each MWC_j **in** $\mathcal{C}$, i$\neq$ j
      if combine(i, j) **then**
        add($\mathcal{N}$, combination(i, j));
        remove($\mathcal{D}$, MWC_i);
        remove($\mathcal{D}$, MWC_j);
  $\mathcal{C} := \mathcal{N}$;
  $\mathcal{D} := \mathcal{D} \bigcup \mathcal{C}$;
until $\mathcal{C} = \emptyset$;

where $\mathcal{D}$ is the set that will contain the results, initially containing all the bigrams; $\mathcal{C}$ - a temporary set currently used in an iteration; and $\mathcal{N}$ - the newly generated n-grams in the current iteration.

The following criterion is considered for combining two multi-word collocations (MWCs) into a larger one: two MWCs can combine if they have at least one term that is different and one that is shared (i.e., that has the same position in text). In the procedure above, the predicate *combine(i, j)* checks whether this criterion is satisfied by the expressions MWC_i and MWC_j, and *combination(i, j)* is the resulting MWC (obtained by merging the terms involved).

At each step, the procedure tries all the possible combinations among already generated MWCs using the above stated composition criterion. It adds the new combinations to the results set $\mathcal{D}$, from which it then eliminates the participating (subsumed) MWCs.

The process is repeated as long as new MWCs can be constructed from the MWCs generated in the previous step. After a finite number of iterations, the procedure terminates since the set of new expressions that can be generated is finite (it is localized within a sentence). It is easy to check that the time complexity of the algorithm is polynomial in the size of the initial bigrams set.

4.2 *Association measures*

The MWCs extracted with the algorithm described above represent all the syntactically bound co-occurrences of terms in the corpus. In order to identify the good collocation candidates among them we proposed 4 methods for quantifying their degree of collocability.

The first method simply computes the MWCs frequency in the corpus. The second uses the log-likelihood values initially assigned to bigrams and considers the sum of participating bigrams' score as a global score for a MWC. The third method tries to find MWCs whose global score is balanced and considers the harmonic mean as an association measure.

Finally, as a fourth method, we apply the log-likelihood test, the same test that FipsCo applied to words in order to score collocation bigrams. We instead apply it to bigrams, in order to score the trigrams build from these bigrams. The contingency table (which is used to compute the log-likelihood values) contains the joint and marginal frequencies for each two bigrams, i.e., the corpus frequency of the two bigrams together (as a trigram), and respectively the corpus frequency of each of the two bigrams.

In order to apply this measure to arbitrarily long MWCs, we can apply it recursively to the sub-MWCs composing a given MWC. Let MWC_1 and MWC_2 be two MWCs that compose a larger MWC (as described in 4.1). The log-likelihood score is computed using a contingency table for the pair (MWC_1, MWC_2), listing co-occurrence frequencies related to each of the two sub-expressions.

5 The experiment. Results and discussion

We applied the method of identifying multi-word collocations as presented in the previous section on a corpus of 948 English articles from the magazine "The Economist". The collection of texts, amounting to about 870,000 words, was first parsed and about 142,000 syntactically bound bigrams were extracted with FipsCo (no frequency filter was applied). About 7.00% of the extracted bigrams involved more than two words[4].

We then extracted trigrams using the word linking method presented in subsection 4.1. We obtained a number of 54,888 trigrams, divided in 13,990, 27,121, and 13,777 for each pivot position case (i.e., left, middle, and right respectively). Table 1 shows the 10 most frequent trigrams in the whole set, and the top 10 trigrams according to the log-likelihood measure.

trigram	**freq**	**trigram**	**log**
weapon of mass destruction	38	weapon of mass destruction	579.03
have impact on	17	have impact on	214.35
go out of	15	move from to	126.10
pull out of	14	turn blind eye	124.01
make difference to	11	rise from in	120.57
rise in to	10	play role in	110.07
move from to	10	make difference to	109.46
rise from in	10	rise in to	105.43
play role in	9	second world war	105.42
be to in	8	rise from to	99.08

Table 1: *Top 10 trigrams according to frequency and log-likelihood*

We consider that both the frequency and the log-likelihood measures are appropriate for scoring collocations, with the log-likelihood slightly more precise.

The measure based on the sum yields uninteresting results, as an expression may obtain a good score if it happen to contain a top scored bigram (as "*prime minister*"), even if it is not a collocation (e.g., "*prime minister promise*").

The measure based on the harmonic mean allows for the identification of good multi-word collocations, like "weapons of mass destruction" that received the best score. Still, we judge its results less satisfactory than those obtained with the log-likelihood measure.

However, to estimate the efficiency of these measures a solid evaluation should be performed against a gold standard, possibly by adopting a n-best strategy, as in (Krenn & Evert 2001).

[4] FipsCo is able to extract some multi-word collocations as bigrams involving a compound, idiom or collocation already present in the lexicon.

We were interested in the syntactic configurations of the multi-word collocations obtained, as they could suggest syntactic patterns to use for the extraction of multi-word collocations directly from parsed text. The most frequent association types found in the corpus are listed in Table 2, together with an example for each type[5].

rel1	**rel2**	**frequency**	**example**
Noun-Prep-Noun	Adjective-Noun	5607	round of presidential election
Verb-Object	Verb-Prep	5364	have impact on
Subject-Verb	Verb-Prep	4904	share fall by
Subject-Verb	Verb-Object	4659	budget face shortfall
Verb-Object	Adjective-Noun	4622	turn blind eye
Adjective-Noun	Subject-Verb	3834	main reason be
Verb-Prep	Verb-Prep	3232	move from to
Verb-Object	Compound	2366	declare state of emergency
Verb-Object	Subject-Verb	1693	want thing be
Noun-Noun	Noun-Prep-Noun	1627	world standard of prosperity

Table 2: *The 10 most frequent association types for trigrams*

As mentioned earlier, during the extraction no predefined syntactic patterns were used (we imposed no restriction on the configuration of newly built expressions). Nevertheless, the trigram patterns which are discovered are dependent on the bigram patterns used by FipsCo extraction system, therefore their coverage depend on how exhaustive the initial patterns are.

6 Conclusions and future work

We have presented a method for the multi-word collocation extraction that relies on the previous extraction of collocation bigrams from text, and is based on iteratively associating already constructed collocations using a syntactic criterion. We have used several measures for estimating the strength of the association. In particular, we applied the log-likelihood ratio statistical test (initially used for word bigrams) to the extracted multi-word collocations. This test appears to be, together with the frequency, the relatively best measure for evaluating the collocational strength.

The methodology used is based on a hybrid (linguistic and statistical) approach aimed at improving the coverage and the precision of multi-word collocation extraction. Unlike purely statistical approaches, the method presented can handle collocations whose terms occur in text at a long distance from each other

[5] The frequency counts refer to the distinct collocations extracted, and do not take into account how many instances a given collocation may have in the corpus.

because of to the various syntactic transformations the collocations can undergo. At the same time, the results are grammatical, due to the syntactically based filter of candidates and to the syntactic nature of the criterion used for the composition of longer multi-word collocations.

Another important advantage over the multi-word collocation extraction methods ignoring the text syntactic structure is that there is no limitation on the length of candidates that can be build. Classical methods (Choueka et al. 1983) were forced to limit to 6-word collocations, and even more recent methods that do not apply a syntactic filter recognize the same limit (Dias 2003).

Further developments of the method include finding criteria for the delimitation of *n*-grams within the sentence, that is, for settling a limit between subsumed and subsuming collocations.

The method will be integrated into a concordance and alignment system (Nerima et al. 2003), which will allow the visualization of extracted multi-word collocations in the source text, and in the parallel (translated) text, when available.

Acknowledgements. This work was carried out in the framework of the research project "Linguistic Analysis and Collocation Extraction" adopted by the Geneva International Academic Network (GIAN) for 2002-2003.

REFERENCES

Ballim, Afzal & Vincenzo Pallotta, eds. 2002. *Robust Methods in Analysis of Natural Language Data* (= Special Issue of *Natural Language Engineering*, 8:2/3). Cambridge: Cambridge University Press.

Benson, Morton, Evelyn Benson & Robert Ilson. 1986. *The BBI Dictionary of English Word Combinations*. Amsterdam: John Benjamins.

Benson, Morton. 1990. "Collocations and General-Purpose Dictionaries". *International Journal of Lexicography* 3:1.23-35.

Choueka, Yaacov, S. T. Klein & E. Neuwitz. 1983. "Automatic Retrieval of Frequent Idiomatic and Collocational Expressions in a Large Corpus". *Journal of the Association for Literary and Linguistic Computing* 4:1.34-38.

Church, Kenneth W. & Patrick Hanks. 1990. "Word Association Norms, Mutual Information and Lexicography". *Computational Linguistics* 16:1.22-29.

Dial, Gaël. 2003. "Multiword Unit Hybrid Extraction". *Proceedings of the Workshop on Multiword Expressions at the 41th Annual Meeting of the Association for Computational Linguistics* (*ACL'03*), 41-48. Sapporo, Japan.

Dunning, Ted. 1993. "Accurate Methods for the Statistics of Surprise and Coincidence". *Computational Linguistics* 19:1.61-74.

Evert, Stefan & Brigitte Krenn. 2001. "Methods for the Qualitative Evaluation of Lexical Association Measures". *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 188-195. Toulouse, France.

Firth, John R. 1957. "Modes of meaning". *Papers in Linguistics* ed. by J. R. Firth, 190-215. Oxford: Oxford University Press.

Goldman, Jean-Philippe, Luka Nerima & Eric Wehrli. 2001. "Collocation Extraction Using a Syntactic Parser". *Proceedings of the Workshop on Collocation at the 39th Annual Meeting of the Association for Computational Linguistics* (*ACL'01*), 61-66. Toulouse, France.

Grishman, Ralph & John Sterling. 1994. "Generalizing Automatically Generated Selectional Patterns". *Proceedings of the 15th International Conference on Computational Linguistics* (*COLING'94*), 742-747. Kyoto, Japan.

Harris, Zelig S. 1988. *Language and Information*. New York: Columbia University Press.

Laenzlinger, Christopher & Eric Wehrli. 1991. "Fips, un analyseur interactif pour le français". *TA Informations* 32:2.35-49.

Lin, Dekang. 1998. "Extracting Collocations from Text Corpora". *First Workshop on Computational Terminology*, 57-63. Montréal, Canada.

Manning, Christopher & Heinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, Mass.: MIT Press.

Mel'cuk, Igor A. et al. 1984, 1988, 1992, 1999. *Dictionnaire explicatif et combinatoire du français contemporain: Recherches lexico-sémantiques I, II, III, IV*. Montréal: Presses de l'Université de Montréal.

Nerima, Luka, Violeta Seretan & Eric Wehrli. 2003. "Creating a Multilingual Collocation Dictionary from Large Text Corpora". *Research Note Sessions of the 10th Conference of the European Chapter of the Association for Computational Linguistics* (*EACL'03*), 131-134. Budapest.

Sag, Ivan, Timothy Baldwin, Francis Bond, Ann Copestake & Dan Flickinger. 2002. "Multiword Expressions: A Pain in the Neck for NLP". *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics* (*CICLING 2002*), 1-15. Mexico City.

Sinclair, John. 1991. *Corpus, Concordance, Collocation*. Oxford: Oxford University Press.

Smadja, Frank. 1993. "Retrieving Collocations form Text: Xtract". *Computational Linguistics* 19:1.143-177.

Combining Independent Modules in Lexical Multiple-Choice Problems

PETER D. TURNEY*, MICHAEL L. LITTMAN**,
JEFFREY BIGHAM*, & VICTOR SHNAYDER**

*NRC, **Rutgers Univ., *Univ. of Washington, **Harvard Univ.

Abstract

Existing statistical approaches to natural language problems are very coarse approximations to the true complexity of language processing. As such, no single technique will be best for all problem instances. Many researchers are examining ensemble methods that combine the output of multiple modules to create more accurate solutions. This paper examines three merging rules for combining probability distributions: the familiar mixture rule, the logarithmic rule, and a novel product rule. These rules were applied with state-of-the-art results to two problems used to assess human mastery of lexical semantics — synonym questions and analogy questions. All three merging rules result in ensembles that are more accurate than any of their component modules. The differences among the three rules are not statistically significant, but it is suggestive that the popular mixture rule is not the best rule for either of the two problems.

1 Introduction

Asked to articulate the relationship between the words broad and road, you might consider a number of possibilities. Orthographically, the second can be derived from the first by deleting the initial letter, while semantically, the first can modify the second to indicate above-average width. Many possible relationships would need to be considered, depending on the context. In addition, many different computational approaches could be brought to bear, leaving a designer of a natural language processing system with some difficult choices. A sound software engineering approach is to develop separate modules using independent strategies, then to combine the output of the modules to produce a unified solver.

The concrete problem we consider here is predicting the correct answers to multiple-choice questions.Each instance consists of a context and a finite set of choices, one of which is correct. Modules produce a probability distribution over the choices and a merging rule is used to combine these distributions into one. This distribution, along with relevant utilities, can then be used to select a candidate answer from the set of choices. The merging rules we considered are parameterized, and we set parameters by a maximum likelihood approach on a collection of training instances.

Many problems can be cast in a multiple-choice framework, including optical digit recognition (choices are the 10 digits), word sense disambiguation (choices are a word's possible senses), text categorization (choices are the classes), and part-of-speech tagging (choices are the grammatical categories). This paper looks at multiple-choice synonym questions (part of the Test of English as a Foreign Language) and multiple-choice verbal analogy questions (part of the SAT college entrance exam).Recent work has shown that algorithms for solving multiple-choice synonym questions can be used to determine the *semantic orientation* of a word; i.e., whether the word conveys praise or criticism (Turney & Littman 2003b). Other research establishes that algorithms for solving multiple-choice verbal analogy questions can be used to ascertain the *semantic relation* in a noun-modifier expression; e.g., in the expression "laser printer", the modifier "laser" is an *instrument* used by the noun "printer" (Turney & Littman 2003a).

The paper offers two main contributions. First, it introduces and evaluates several new modules for answering multiple-choice synonym questions and verbal analogy questions. Second, it presents a novel product rule for combining module outputs and compares it with other similar merging rules.

Section 2 formalizes the problem addressed in this paper and introduces the three merging rules we study in detail: the mixture rule, the logarithmic rule, and the product rule. Section 3 presents empirical results on synonym problems and Section 4 considers analogy problems. Section 5 summarizes and wraps up.

2 Module combination

The following synonym question is a typical multiple-choice question: hidden:: (a) laughable, (b) veiled, (c) ancient, (d) revealed. The stem, hidden, is the question. There are $k = 4$ choices, and the question writer asserts that exactly one (in this case, (b)) has the same meaning as the stem word. The accuracy of a solver is measured by its fraction of correct answers on a set of ℓ testing instances.

In our setup, knowledge about the multiple-choice task is encapsulated in a set of n modules, each of which can take a question instance and return a probability distribution over the k choices. For a synonym task, one module might be a statistical approach that makes judgments based on analyses of word co-occurrence, while another might use a thesaurus to identify promising candidates. These modules are applied to a training set of m instances, producing probabilistic "forecasts"; $p_{ij}^h \geq 0$ represents the probability assigned by module $1 \leq i \leq n$ to choice $1 \leq j \leq k$ on training instance $1 \leq h \leq m$. The estimated probabilities are distributions of the choices for each module i on each instance h: $\sum_j p_{ij}^h = 1$.

2.1 *Merging rules*

The rules we considered are parameterized by a set of weights w_i, one for each module. For a given merging rule, a setting of the weight vector w induces a probability distribution over the choices for any instance. Let $D_j^{h,w}$ be the probability assigned by the merging rule to choice j of training instance h when the weights are set to w. Let $1 \leq a(h) \leq k$ be the correct answer for instance h. We set weights to maximize the likelihood of the training data: $w = \mathrm{argmax}_{w'} \prod_h D_{a(h)}^{h,w'}$. The same weights maximize the *mean likelihood*, the geometric mean of the probabilities assigned to correct answers.

We focus on three merging rules in this paper. The *mixture rule*combines module outputs using a weighted sum and can be written $M_j^{h,w} = \sum_i w_i p_{ij}^h$, where $D_j^{h,w} = M_j^{h,w} / \sum_j M_j^{h,w}$ is the probability assigned to choice j of instance h and $0 \leq w_i \leq 1$. The rule can be justified by assuming each instance's answer is generated by a single module chosen via the distribution $w_i / \sum_i w_i$.

The *logarithmic rule* combines the logarithm of module outputs by $L_j^{h,w} = \exp(\sum_i w_i \ln p_{ij}^h) = \prod_i (p_{ij}^h)^{w_i}$, where $D_j^{h,w} = L_j^{h,w} / \sum_j L_j^{h,w}$ is the probability the rule assigns to choice j of instance h. The weight w_i indicates how to scale the module probabilities before they are combined multiplicatively. Note that modules that output zero probabilities must be modified before this rule can be used.

The *product rule* can be written in the form $P_j^{h,w} = \prod_i (w_i p_{ij}^h + (1 - w_i)/k)$, where $D_j^{h,w} = P_j^{h,w} / \sum_j P_j^{h,w}$ is the probability the rule assigns to choice j. The weight $0 \leq w_i \leq 1$ indicates how module i's output should be mixed with a uniform distribution (or a prior, more generally) before outputs are combined multiplicatively. As with the mixture and logarithmic rules, a module with a weight of zero has no influence on the final assignment of probabilities.

For the experiments reported here, we adopted a straightforward approach to finding the weight vector w that maximizes the likelihood of the data. The weight optimizer reads in the output of the modules, chooses a random starting point for the weights, then hillclimbs using an approximation of the partial derivative. Although more sophisticated optimization algorithms are well known, we found that the simple discrete gradient approach worked well for our application.

2.2 *Related work*

Merging rules of various sorts have been studied for many years, and have gained prominence recently for natural language applications. Use of the mixture rule and its variations is quite common. Recent examples include the work of Brill & Wu (1998) on part-of-speech tagging, Littman et al. (2002) on crossword-puzzle clues and Florian & Yarowsky (2002) on a word-sense disambiguation task. We

use the name "mixture rule" by analogy to the mixture of experts model (Jacobs et al. 1991), which combined expert opinions in an analogous way. In the forecasting literature, this rule is also known as the linear opinion pool; Jacobs (1995) provides a summary of the theory and applications of the mixture rule in this setting.

The logarithmic opinion pool of Heskes (1998) is the basis for our logarithmic rule. Boosting (Schapire 1999) also uses a logistic-regression-like rule to combine outputs of simple modules to perform state-of-the-art classification. The product of experts approach also combines distributions multiplicatively, and Hinton (1999) argues that this is an improvement over the "vaguer" probability judgments commonly resulting from the mixture rule. A survey by Xu et al. (1992) includes the equal-weights version of the mixture rule. A derivation of the unweighted product rule appears in Xu et al. (1992) and Turney et al. (2003).

An important contribution of the current work is the product rule, which shares the simplicity of the mixture rule and the probabilistic justification of the logarithmic rule. We have not seen an analog of this rule in the forecasting or learning literatures.

3 Synonyms

We constructed a training set of 431 4-choice synonym questions and randomly divided them into 331 training questions and 100 testing questions. We created four modules, described next, and ran each module on the training set. We used the results to set the weights for the three merging rules and evaluated the resulting synonym solver on the test set. Module outputs, where applicable, were normalized to form a probability distribution by scaling them to add to one before merging.

3.1 *Modules*

LSA. Following Landauer & Dumais (1997), we used latent semantic analysis to recognize synonyms. Our LSA module queried the web interface developed at the University of Colorado (`lsa.colorado.edu`), which has a 300-dimensional vector representation for each of tens of thousands of words.

PMI-IR. Our Pointwise Mutual Information-Information Retrieval module used the AltaVista search engine (`www.altavista.com`) to determine the number of web pages that contain the choice and stem in close proximity. PMI-IR used the third scoring method (near each other, but not near `not`) designed by Turney (2001), since it performed best in this earlier study.

Thesaurus. Our Thesaurus module also used the web to measure word pair similarity. The module queried Wordsmyth (`www.wordsmyth.net`) and collected any words listed in the "Similar Words", "Synonyms", "Crossref. Syn.",

Synonym solvers	Accuracy	Mean likelihood
LSA only	43.8%	.2669
PMI-IR only	69.0%	.2561
Thesaurus only	69.6%	.5399
Connector only	64.2%	.3757
All: mixture	80.2%	.5439
All: logarithmic	82.0%	.5977
All: product	80.0%	.5889

Table 1: *Comparison of results for merging rules on synonym problems*

and "Related Words" fields. The module created synonym lists for the stem and for each choice, then scored them by their overlap.

Connector. Our Connector module used summary pages from querying Google (`google.com`) with pairs of words to estimate pair similarity. It assigned a score to a pair of words by taking a weighted sum of both the number of times they appear separated by one of the symbols **[**, **"**, **:**, **,**, =, /, \, **(**, **]**, means, defined, equals, synonym, whitespace, and and and the number of times dictionary or thesaurus appear anywhere in the Google summaries.

3.2 *Results*

Table 1 presents the result of training and testing each of the four modules on synonym problems. The first four lines list the accuracy and mean likelihood obtained using each module individually (using the product rule to set the individual weight). The highest accuracy is that of the Thesaurus module at 69.6%. All three merging rules were able to leverage the combination of the modules to improve performance to roughly 80% — statistically significantly better than the best individual module.

Although the accuracies of the merging rules are nearly identical, the product and logarithmic rules assign higher probabilities to correct answers, as evidenced by the mean likelihood. To illustrate the decision-theoretic implications of this difference, imagine using the probability judgments in a system that receives a score of $+1$ for each right answer and $-1/2$ for each wrong answer, but can skip questions. In this case, the system should make a guess whenever the highest probability choice is above $1/3$. For the test questions, this translates to scores of 71.0 and 73.0 for the product and logarithmic rules, but only 57.5 for the mixture rule; it skips many more questions because it is insufficiently certain.

3.3 *Related work and discussion*

Landauer & Dumais (1997) introduced the Test of English as a Foreign Language (TOEFL) synonym task as a way of assessing the accuracy of a learned

Reference	Accuracy	95% confidence
Landauer & Dumais (1997)	64.40%	52.90–74.80%
non-native speakers	64.50%	53.01–74.88%
Turney (2001)	73.75%	62.71–82.96%
Jarmasz & Szpakowicz (2003)	78.75%	68.17–87.11%
Terra & Clarke (2003)	81.25%	70.97–89.11%
Product rule	97.50%	91.26–99.70%

Table 2: *Published TOEFL synonym results*

representation of lexical semantics. Several studies have since used the same data set for direct comparability; Table 2 presents these results.

The accuracy of LSA (Landauer & Dumais 1997) is statistically indistinguishable from that of a population of non-native English speakers on the same questions. PMI-IR (Turney 2001) performed better, but the difference is not statistically significant. Jarmasz & Szpakowicz (2003) give results for a number of relatively sophisticated thesaurus-based methods that looked at path length between words in the heading classifications of Roget's Thesaurus. Their best scoring method was a statistically significant improvement over the LSA results, but not over those of PMI-IR. Terra & Clarke (2003) studied a variety of corpus-based similarity metrics and measures of context and achieved a statistical tie with PMI-IR and the results from Roget's Thesaurus.

To compare directly to these results, we removed the 80 TOEFL instances from our collection and used the other 351 instances for training the product rule. The resulting accuracy was statistically significantly better than all previously published results, even though the individual modules performed nearly identically to their published counterparts.

4 Analogies

Synonym questions are unique because of the existence of thesauri — reference books designed precisely to answer queries of this form. The relationships exemplified in analogy questions are quite a bit more varied and are not systematically compiled. For example, the analogy question cat:meow:: (a) mouse:scamper, (b) bird:peck, (c) dog:bark, (d) horse:groom, (e) lion:scratch requires that the reader recognize that (c) is the answer because both (c) and the stem are examples of the relation "X is the name of the sound made by Y". This type of common sense knowledge is rarely explicitly documented.

In addition to the computational challenge they present, analogical reasoning is recognized as an important component in cognition, including language comprehension (Lakoff & Johnson 1980) and high level perception (Chalmers et al. 1992).

To study module merging for analogy problems, we collected 374 5-choice instances. We randomly split the collection into 274 training instances and 100 testing instances.We next describe the novel modules we developed for attacking analogy problems and present their results.

4.1 *Modules*

Phrase vectors. We wish to score candidate analogies of the form A:B::C:D (A is to B as C is to D). The quality of a candidate analogy depends on the similarity of the relation R_1 between A and B to the relation R_2 between C and D. The relations R_1 and R_2 are not given to us; the task is to infer these relations automatically. Our approach to this task is to create vectors r_1 and r_2 that represent features of R_1 and R_2, and then measure the similarity of R_1 and R_2 by the cosine of the angle between the vectors r_1 and r_2 (Turney & Littman 2003a). We create a vector, r, to characterize the relationship between two words, X and Y, by counting the frequencies of 128 different short phrases containing X and Y. Phrases include "X for Y", "Y with X", "X in the Y", and "Y on X". We use these phrases as queries to AltaVista and record the number of hits (matching web pages) for each query. This process yields a vector of 128 numbers for a pair of words X and Y. The resulting vector r is a kind of *signature* of the relationship between X and Y.

Thesaurus paths. Another way to characterize the semantic relationship, R, between two words, X and Y, is to find a path through a thesaurus or dictionary that connects X to Y or Y to X. In our experiments, we used the WordNet thesaurus (Fellbaum 1998). We view WordNet as a directed graph and the Thesaurus Paths module performed a breadth-first search for paths from X to Y or Y to X. For a given pair of words, X and Y, the module considers all shortest paths in either direction up to three links. It scores the candidate analogy by the maximum degree of similarity between any path for A and B and any path for C and D. The degree of similarity between paths is measured by their number of shared features.

Lexical relation modules. We implemented a set of more specific modules using the WordNet thesaurus. Each module checks if the stem words match a particular relationship in the database. If they do not, the module returns the uniform distribution. Otherwise, it checks each choice pair and eliminates those that do not match. The relations tested are: **Synonym**, **Antonym**, **Hypernym**, **Hyponym**, **Meronym:substance**, **Meronym:part**, **Meronym:member**, **Holonym:substance**, and also **Holonym:member**.

Similarity. Dictionaries are a natural source to use for solving analogies because definitions can express many possible relationships and are likely to make the relationships more explicit than they would be in general text. We employed two definition similarity modules: **Similarity:dict** uses `dictionary.com`

Analogy solvers	Accuracy	Mean likelihood
Phrase Vectors	38.2%	.2285
Thesaurus Paths	25.0%	.1977
Synonym	20.7%	.1890
Antonym	24.0%	.2142
Hypernym	22.7%	.1956
Hyponym	24.9%	.2030
Meronym:substance	20.0%	.2000
Meronym:part	20.8%	.2000
Meronym:member	20.0%	.2000
Holonym:substance	20.0%	.2000
Holonym:member	20.0%	.2000
Similarity:dict	18.0%	.2000
Similarity:wordsmyth	29.4%	.2058
all: mixture	42.0%	.2370
all: logarithmic	43.0%	.2354
all: product	45.0%	.2512
no PV: mixture	31.0%	.2135
no PV: logarithmic	30.0%	.2063
no PV: product	37.0%	.2207

Table 3: *Comparison of results for merging rules on analogy problems*

and **Similarity:wordsmyth** uses `wordsmyth.net` for definitions. Each module treats a word as a vector formed from the words in its definition. Given a potential analogy A:B::C:D, the module computes a vector similarity of the first words (A and C) and adds it to the vector similarity of the second words (B and D).

4.2 *Results*

We ran the 13 modules described above on our set of training and testing analogy instances, with the results appearing in Table 3 (the product rule was used to set weights for computing individual module mean likelihoods). For the most part, individual module accuracy is near chance level (20%), although this is misleading because most of these modules only return answers for a small subset of instances. Some modules did not answer a single question on the test set. The most accurate individual module was the search-engine-based Phrase Vectors (PV) module. The results of merging all modules was only a slight improvement over PV alone, so we examined the effect of retraining without the PV module. The product rule resulted in a large improvement (though not statistically significant) over the best remaining individual module (37.0% vs. 29.4% for Similarity:wordsmyth).

We once again examined the result of deducting $1/2$ point for each wrong

answer. The full set of modules scored 31, 33, and 43 using the mixture, logarithmic, and product rules. As in the synonym problems, the logarithmic and product rules assigned probabilities more precisely. In this case, the product rule appears to have a major advantage.

5 Conclusion

We applied three trained merging rules to a set of multiple-choice problems and found all were able to produce state-of-the-art performance on a standardized synonym task by combining four less accurate modules. Although all three rules produced comparable accuracy, the popular mixture rule was consistently weaker than the logarithmic and product rules at assigning high probabilities to correct answers. We provided first results on a challenging verbal analogy task with a set of novel modules that use both lexical databases and statistical information.

In nearly all the tests that we ran, the logarithmic rule and our novel product rule behaved similarly, with a hint of an advantage for the product rule. One point in favor of the logarithmic rule is that it has been better studied so its theoretical properties are better understood. It also is able to "sharpen" probability distributions, which the product rule cannot do without removing the upper bound on weights. On the other hand, the product rule is simpler, executes much more rapidly, and is more robust in the face of modules returning zero probabilities. We feel the strong showing of the product rule proves it worthy of further study.

Acknowledgements. This research was supported in part by a grant from NASA. We'd like to thank the students and TAs of Princeton's COS302 in 2001 for their pioneering efforts in solving analogy problems; Douglas Corey Campbell, Brandon Braunstein, and Paul Simbi, who provided the first version of our Thesaurus module; Yann LeCun for scanning help, and Haym Hirsh, Philip Resnik, Rob Schapire, Matthew Stone, and Kagan Tumer who served as sounding boards on the topic of module merging.

REFERENCES

Brill, Eric & Jun Wu. 1998. "Classifier Combination for Improved Lexical Disambiguation". *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (*ACL'98*), vol. 1, 191-195. Montréal, Canada.

Chalmers, David J., Robert M. French, & Douglas R. Hofstadter. 1992. "High-level Perception, Representation and Analogy: A Critique of Artificial Intelligence Methodology". *Journal of Experimental and Theoretical Artificial Intelligence* 4:185-211.

Fellbaum, Christiane. 1998. *WordNet: An Electronic Lexical Database*. Cambridge, Mass.: MIT Press.

Florian, Radu & David Yarowsky. 2002. "Modeling Consensus: Classifier Combination for Word Sense Disambiguation". *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 25-32. Philadelphia.

Heskes, Tom. 1998. "Selecting Weighting Factors in Logarithmic Opinion Pools". *Advances in Neural Information Processing Systems* 10:266-272.

Hinton, Geoffrey E. 1999. "Products of Experts". *Proceedings of the 9th International Conference on Artificial Neural Networks* (*ICANN 99*), 1:1-6. Edinburgh, Scotland.

Jacobs, Robert A. 1995. "Methods for Combining Experts' Probability Assessments". *Neural Computation* 7:5.867-888.

Jacobs, Robert A., Michael I. Jordan, Steve J. Nowlan & Geoffrey E. Hinton. 1991. "Adaptive Mixtures of Experts". *Neural Computation* 3:79-87.

Jarmasz, Mario & Stan Szpakowicz. 2003. "Roget's Thesaurus and Semantic Similarity". *Proceedings of the International Conference on Recent Advances in Natural Language Processing* (*RANLP-03*), 212-219. Borovets, Bulgaria.

Lakoff, George & Mark Johnson. 1980. *Metaphors We Live By*. Chicago: University of Chicago Press.

Landauer, Thomas K. & Susan T. Dumais. 1997. "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge". *Psychological Review* 104:2.211-240.

Littman, Michael L., Greg A. Keim & Noam Shazeer. 2002. "A Probabilistic Approach to Solving Crossword Puzzles". *Artificial Intelligence* 134:23-55.

Schapire, Robert E. 1999. "A Brief Introduction to Boosting". *Proceedings of the 16th International Joint Conference on Artificial Intelligence* (*IJCAI-99*), 1401-1406. Stockholm, Sweden.

Terra, Egidio & C. L. A. Clarke. 2003. "Frequency Estimates for Statistical Word Similarity Measures". *Proceedings of the Human Language Technology and North American Chapter of Association of Computational Linguistics Conference 2003* (*HLT/NAACL 2003*), 244-251. Edmonton, Alberta, Canada.

Turney, Peter D. 2001. "Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL". *Proceedings of the 12th European Conference on Machine Learning* (*ECML-2001*), 491-502. Freiburg, Germany.

Turney, Peter D. & Michael L. Littman. 2003a. "Learning Analogies and Semantic Relations". Technical Report ERB-1103. Ottawa, Ontario, Canada: National Research Council, Institute for Information Technology.

Turney, Peter D. & Michael L. Littman. 2003b. "Measuring Praise and Criticism: Inference of Semantic Orientation from Association". *ACM Transactions on Information Systems* 21:4.315-346.

Turney, Peter D., Michael L. Littman, Jeffrey Bigham & Victor Shnayder. 2003. "Combining Independent Modules to Solve Multiple-Choice Synonym and Analogy Problems". *Proceedings of the International Conference on Recent Advances in Natural Language Processing* (*RANLP-03*), 482-489. Borovets, Bulgaria.

Xu, Lei, Adam Krzyzak & Ching Y. Suen. 1992. "Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition". *IEEE Transactions on Systems, Man and Cybernetics* 22:3.418-435.

Roget's Thesaurus and Semantic Similarity[1]

MARIO JARMASZ[2] & STAN SZPAKOWICZ

School of Information Technology and Engineering, University of Ottawa

Abstract

We have implemented a system that measures semantic similarity using a computerized 1987 *Roget's Thesaurus*, and evaluated it by performing a few typical tests. We compare the results of these tests with those produced by *WordNet*-based similarity measures. One of the benchmarks is Miller & Charles' list of 30 noun pairs that human judges had rated for similarity. We correlate these ratings with those computed by several NLP systems. The 30 pairs can be traced back to Rubenstein & Goodenough's 65 pairs that we have also studied. Our *Roget's*-based system gets correlations of .878 for the smaller and .818 for the larger list of noun pairs. We further evaluate our measure by using *Roget's* and *WordNet* to answer *TOEFL*, *ESL* and *Reader's Digest* questions: the correct synonym must be selected amongst a group of four words. Our system gets right 78.75%, 82.00% and 74.33% of the questions respectively.

1 Introduction

People identify synonyms — strictly speaking, near-synonyms (Edmonds & Hirst 2002) — such as *angel–cherub*, without being able to define synonymy properly. The term tends to be used loosely, even in the crucially synonymy-oriented *WordNet* with the synset as the basic semantic unit (Fellbaum 1998:23). Miller & Charles (1991) restate a formal, and linguistically quite inaccurate, definition of synonymy usually attributed to Leibniz: "two words are said to be synonyms if one can be used in a statement in place of the other without changing the meaning of the statement". With this strict definition there may be no perfect synonyms in natural language (Edmonds & Hirst, *ibid*.). For NLP systems it is often more useful to establish the degree of synonymybetween two words, referred to as *semantic similarity*.

Miller & Charles' semantic similarity is a continuous variable that describes the degree of synonymy between two words (*ibid*.). They argue that native speakers can order pairs of words by semantic similarity, for example *ship–vessel*, *ship–watercraft*, *ship–riverboat*, *ship–sail*, *ship–house*, *ship–dog*, *ship–sun*. The concept can be usefully extended to quantify relations between non-synonymous but closely related words, such as *airplane–wing*.

[1] A longer version appeared in the Proceedings of the International Conference on Recent Advances in Natural Language Processing, Sept. 2003, 212-219. Borovets, Bulgaria.

[2] Currently at the National Research Council of Canada.

Rubenstein & Goodenough (1965) investigated the validity of the assumption that "... pairs of words which have many contexts in common are semantically closely related". With the help of human experts, they established *synonymy judgments* for 65 noun pairs. Miller & Charles (*ibid*.) selected 30 of those pairs, and studied semantic similarity as a function of the contexts in which words are used. Others have calculated similarity using semantic nets (Rada et al. 1989), in particular *WordNet* (Resnik 1995, Jiang & Conrath 1997, Lin 1998, Hirst & St-Onge 1998, Leacock & Chodorow 1998) and *Roget's Thesaurus* (McHale 1998), or statistical methods (Landauer & Dumais 1997, Turney 2001, Turney et al. 2003).

We set out to test the intuition that *Roget's Thesaurus*, sometimes treated as a book of synonyms,allows us to measure semantic similarity effectively. We demonstrate some of *Roget's* qualities that make it a realistic alternative to *WordNet*, in particular for the task of measuring semantic similarity. We propose a measure of *semantic distance*, the inverse of semantic similarity (Budanitsky & Hirst 2001) based on *Roget's* taxonomy. We convert it into a semantic similarity measure, and empirically compare to human judgments and to those of NLP systems. We consider the tasks of assigning a similarity value to pairs of nouns and choosing the correct synonym of a problem word given the choice of four target words. We explain in detail the measures and the experiments, and draw a few conclusions.

2 *Roget's Thesaurus* relations as a measure of semantic distance

Resnik (1995) claims that a natural way of calculating semantic similarity in a taxonomy is to measure the distance between the nodes that correspond to the items we compare: the shorter the path, the more similar the items. Given multiple paths, we take the shortest. Resnik states a widely acknowledged problem with edge counting. It relies on the notion that links in the taxonomy represent uniform distances, and it is therefore not the best semantic distance measure for *WordNet*. We want to investigate this claim for *Roget's*, as its hierarchy is very regular.

Roget's Thesaurus has many advantages. It is based on a well-constructed concept classification, and its entries were written by professional lexicographers. It contains around 250,000 words compared to *WordNet's* almost 200,000. *Roget's* does not have some of *WordNet's* shortcomings, such as the lack of links between parts of speech and the absence of topical groupings. The clusters of closely related words are obviously not the same in both resources. *WordNet* relies on a set of about 15 semantic relations. Search in this lexical database requires a word and a semantic relation; for every word some (but never all) of 15 relations can be used in search. It is impossible to express a relationship that involves more than one of the 15 relations: it cannot be stored in *WordNet*. The *Thesaurus* can link the noun *bank*, the business that provides financial ser-

vices, and the verb *invest*, to give money to a bank to get a profit, as used in the following sentences, by placing them in a common head ***784*** *Lending*.

(1) Mary went to the *bank* yesterday.

(2) She *invested* $5,000.00 in mutual funds.

This type of connection cannot be described using *WordNet's* semantic relations. While an English speaker can identify a relation not provided by *WordNet*, for example that one invests money in a bank, this is not sufficient for use in computer systems.

We used a computerized version of the 1987 edition of Penguin's *Roget's Thesaurus of English Words and Phrases* (Jarmasz & Szpakowicz 2001) to calculate the semantic distance. *Roget's* structure allows an easy implementation of edge counting. Given two words, we look up in the index their references that point into the *Thesaurus*. Next, we find all paths between references in *Roget's* taxonomy. On another version of *Roget's*, McHale (1998) showed that edge counting is a good semantic distance measure.

Eight Classes head this taxonomy. The first three, *Abstract Relations*, *Space* and *Matter*, cover the external world. The remaining ones, *Formation of ideas*, *Communication of ideas*, *Individual volition*, *Social volition*, *Emotion*, *Religion and Morality* deal with the internal world of human beings. A path in *Roget's* ontology always begins with one of the Classes. It branches to one of the 39 Sections, then to one of the 79 Sub-Sections, then to one of the 596 Head Groups and finally to one of the 990 Heads. Each Head is divided into paragraphs grouped by parts of speech: nouns, adjectives, verbs and adverbs. Finally a paragraph is divided into semicolon groups of semantically closely related words. Jarmasz & Szpakowicz (*ibid.*) give a detailed account of *Roget's* structure.

The distance equals the number of edges in the shortest path. Path lengths vary from 0 to 16. Shorter paths mean more closely related words and phrases. As an example, the *Roget's* distance between *feline* and *lynx* is 2. The word ***feline*** has these references:

(1) *animal 365 ADJ.*

(2) *cat 365 N.*

(3) *cunning 698 ADJ.*

The word ***lynx*** has these references:

(1) *cat 365 N.*

(2) *eye 438 N.*

The shortest and longest path are (T is the top of the taxonomy):

- *feline* → *cat* ← *lynx* (the same paragraph)
- *feline* → *cunning* → *ADJ.* → *698. Cunning* → *[698, 699]* → *Complex* → *Section three : Voluntary action* → *Class six : Volition: individual volition* → T ← *Class three : Matter* ← *Section three : Organic matter* ← *Sensation* ← *[438, 439, 440]* ← *438. Vision* ← *N.* ← *eye* ← *lynx*

We convert distance to similarity by subtracting the path length from the maximal path length (Resnik 1995):

sim (w_1, w_2) = 16 - [min distance(r_1, r_2)] (1)

where r_1, r_2 are the sets of references for the words or phrases w_1, w_2.

3 Evaluation based on human judgment

3.1 *The data*

Rubenstein & Goodenough (1965) established *synonymy judgments* for 65 pairs of nouns. They invited 51 judges who assigned to every pair a score between 4.0 and 0.0 indicating semantic similarity. They chose words from non-technical every day English. They felt that, since the phenomenon under investigation was a general property of language, it was not necessary to study technical vocabulary. Miller & Charles (1991) repeated the experiment restricting themselves to 30 pairs of nouns selected from Rubenstein & Goodenough's list, divided equally amongst words with high, intermediate and low similarity.

We repeated both experiments using the *Roget's Thesaurus* system. We decided to compare our results to six other similarity measures that rely on *WordNet*. Pedersen's *WordNet::Similarity Perl Module* (2003) was applied to *WordNet 1.7.1*. The first *WordNet* measure used is edge counting. It serves as a baseline, as it is the simplest and most intuitive measure. The next measure, from Hirst & St-Onge (1998), relies on the path length as well as the number of changes of direction in the path; these changes are defined in function of *WordNet* semantic relations. Jiang & Conrath (1997) propose a combined approach based on edge counting enhanced by the node-based approach of the information content calculation proposed by Resnik (1995). Leacock & Chodorow (1998) count the path length in nodes rather than links, and adjust it to take into account the maximum depth of the taxonomy. Lin (1998) calculates semantic similarity using a formula derived from information theory. Resnik (1995) calculates the information content of the concepts that subsume them in the taxonomy.

We calculate the Pearson product-moment correlation coefficient between the human judgments and the values achieved by the systems. The correlation is significant at the 0.01 level. Tables[1] 1 and 2 contain the results.

3.2 *The results*

We first analyze the results obtained using *Roget's*. The Miller & Charles data in Table 1 show that pairs of words with a score of 16 have high similarity, those with a score of 12-14 have intermediate similarity, and those with a score below 10 are of low similarity. This is intuitively correct, as words or phrases that are

[1] The abbreviations used in the tables stand for: Miller & Charles (MC), Rubenstein & Goodenough (RG), Penguin's *Roget's* (RT), *WordNet* Edges (WN), Hirst & St. Onge (HSO), Jiang & Conrath (JCN), Leacock & Chodorow (LCH), Lin (LIN), Resnik (RES).

Noun Pair	MC	RT	WN	HSO	JCN	LCH	LIN	RES
car–automobile	3.920	16.000	30.000	16.000	1.000	3.466	1.000	6.340
gem–jewel	3.840	16.000	30.000	16.000	1.000	3.466	1.000	12.886
journey–voyage	3.840	16.000	29.000	4.000	0.169	2.773	0.699	6.057
boy–lad	3.760	16.000	29.000	5.000	0.231	2.773	0.824	7.769
coast–shore	3.700	16.000	29.000	4.000	0.647	2.773	0.971	8.974
asylum–madhouse	3.610	16.000	29.000	4.000	0.662	2.773	0.978	11.277
magician–wizard	3.500	14.000	30.000	16.000	1.000	3.466	1.000	9.708
midday–noon	3.420	16.000	30.000	16.000	1.000	3.466	1.000	10.584
furnace–stove	3.110	14.000	23.000	5.000	0.060	1.386	0.238	2.426
food–fruit	3.080	12.000	23.000	0.000	0.088	1.386	0.119	0.699
bird–cock	3.050	12.000	29.000	6.000	0.159	2.773	0.693	5.980
bird–crane	2.970	14.000	27.000	5.000	0.139	2.079	0.658	5.980
tool–implement	2.950	16.000	29.000	4.000	0.546	2.773	0.935	5.998
brother–monk	2.820	14.000	29.000	4.000	0.294	2.773	0.897	10.489
lad–brother	1.660	14.000	26.000	3.000	0.071	1.856	0.273	2.455
crane–implement	1.680	0.000	26.000	3.000	0.086	1.856	0.394	3.443
journey–car	1.160	12.000	17.000	0.000	0.075	0.827	0.000	0.000
monk–oracle	1.100	12.000	23.000	0.000	0.058	1.386	0.233	2.455
cemetery–woodland	0.950	6.000	21.000	0.000	0.049	1.163	0.067	0.699
food–rooster	0.890	6.000	17.000	0.000	0.063	0.827	0.086	0.699
coast–hill	0.870	4.000	26.000	2.000	0.148	1.856	0.689	6.378
forest–graveyard	0.840	6.000	21.000	0.000	0.050	1.163	0.067	0.699
shore–woodland	0.630	2.000	25.000	2.000	0.056	1.674	0.124	1.183
monk–slave	0.550	6.000	26.000	3.000	0.063	1.856	0.247	2.455
coast–forest	0.420	6.000	24.000	0.000	0.055	1.520	0.121	1.183
lad–wizard	0.420	4.000	26.000	3.000	0.068	1.856	0.265	2.455
chord–smile	0.130	0.000	20.000	0.000	0.066	1.068	0.289	2.888
glass–magician	0.110	2.000	23.000	0.000	0.056	1.386	0.123	1.183
rooster–voyage	0.080	2.000	11.000	0.000	0.044	0.470	0.000	0.000
noon–string	0.080	6.000	19.000	0.000	0.052	0.981	0.000	0.000
Correlation	1.000	0.878	0.732	0.689	0.695	0.821	0.823	0.775

Table 1: *Semantic similarity measures using the Miller & Charles data*

in the same semicolon group will have a similarity score of 16, those that are in the same paragraph, part-of-speech or Head will have a score of 10 to 14, and words that cannot be found in the same Head, therefore do not belong to the same concept, will have a score between 0 and 8. *Roget's* results correlate very well with human judgment for the Miller & Charles list ($r = .878$), almost attaining the upper bound ($r = .885$) set by human judges (Resnik 1995) despite the outlier *crane–implement*, two words that have nothing in common in the *Thesaurus*.

The correlation between human judges and *Roget's* for the Rubenstein &

Goodenough data is also very good ($r = .818$) as shown in Table 2. Although we do not present the 65 pairs of words in the list, the outliers merit discussion. *Roget's* deems five pairs of low similarity words to be of intermediate similarity — all with the semantic distance 12. These pairs of words are therefore all found in the same Head and belong to noun groups. The *Roget's* associations are correct but not the most intuitive: *glass–jewel* is assigned a value of 1.78 by the human judges but can be found under the Head ***844*** *Ornamentation*, *car–journey* is assigned 1.55 and is found under the Head ***267*** *Land travel*, *monk–oracle* 0.91 found under Head ***986*** *Clergy*, *boy–rooster* 0.44 under Head ***372*** *Male*, and *fruit–furnace* 0.05 under Head ***301*** *Food: eating and drinking*.

	RG	RT	WN	HSO	JCN	LCH	LIN	RES
Correlation	1.000	0.818	0.787	0.732	0.731	0.852	0.834	0.800

Table 2: *Similarity measures using the Rubenstein & Goodenough data*

Resnik (*ibid*.) argues that edge counting using *WordNet* 1.4 is not a good measure of semantic similarity as it relies on the notion that links in the taxonomy represent uniform distances. Tables 1 and 2 show that this measure performs well for *WordNet* 1.7.1 . This could be explained by the substantial improvement in *WordNet* , including more uniform distances between words.

4 Evaluation based on synonymy problems

4.1 *The data*

Another method of evaluating semantic similarity metrics is to see how well a computer system can score on a standardized synonym test. Such tests have questions where the correct synonym is one of four possible choices. This type of questions can be found in the Test of English as a Foreign Language [TOEFL] (Landauer & Dumais 1997) and English as a Second Language tests [ESL] (Turney 2001), as well as the Reader's Digest Word Power Game [RDWP] (Lewis 2000-2001). Although this evaluation method is not widespread in Computational Linguistics, it has been used in Psychology (Landauer & Dumais, *ibid*.) and Machine Learning (Turney, *ibid*.). In this experiment we use 80 TOEFL, 50 ESL and 300 RDWP questions.

A RDWP question is presented like this: "Check the word or phrase you believe is nearest in meaning. **ode** - A: heavy debt. B: poem. C: sweet smell. D: surprise." (Lewis 2001, n. 938). Our system calculates the semantic distance between the problem word and each choice word or phrase. The choice word with the shortest semantic distance becomes the solution. Choosing the word or phrase that has the most paths with the shortest distance breaks ties. Phrases not found in the *Thesaurus* present a special problem. We calculate the distance

between each word in the choice phrase and the problem word; the conjunction *and*, the preposition *to*, the verb *be* are ignored. The shortest distance between the individual words and the problem word is considered as the semantic distance for the phrase. This technique, although simplistic, lets us deal with phrases like *rise and fall*, *to urge* and *be joyous* that may not be found in the *Thesaurus* as presented. The *Roget's* system is not restricted to nouns when finding the shortest path — nouns, adjectives, verbs and adverbs are all considered.

We put the *WordNet* semantic similarity measures to the same task of answering the synonymy questions. The purpose of our experiment was not to improve the measures, but to use them as a comparison for the *Roget's* system. We choose as the answer the choice word that has the largest semantic similarity value with the problem word. When ties occur, a partial score is given; .5 if two words are tied for the highest similarity value, .33 if three, and .25 if four. The results appear in Table 3. We did not tailor the *WordNet* measures to the task of answering these questions. All of them, except Hirst & St-Onge, rely on the IS-A hierarchy to calculate the path between words. The measures have been limited to finding similarities between nouns, as the *WordNet* hyponym tree only exists for nouns and verbs; there are hardly any links between parts of speech. We did not implement any special techniques to deal with phrases. It is therefore quite probable that the similarity measures can be improved for the task of answering synonymy questions.

We also compare our results to those achieved by state-of-the-art statistical techniques. Latent Semantic Analysis [LSA] is a general theory of acquired similarity and knowledge representation (Landauer & Dumais 1997). It was used to answer the 80 TOEFL questions. Another algorithm, called PMI-IR (Turney 2001), uses Pointwise Mutual Information [PMI] and Information Retrieval [IR] to measure the similarity of pairs of words. It has been evaluated using the TOEFL and ESL questions. Turney et al. (2003) combine four statistical methods, including LSA and PMI-IR, to measure semantic similarity and evaluate it on the same 80 questions.

4.2 *The results*

The *Roget's Thesaurus* system answers 78.75% of the TOEFL questions (Table 3). The two next best systems are Hirst & St-Onge and PMI-IR, which answer 77.91% and 73.75% of the questions respectively. LSA is not too far behind, with 64.38%. Turney et al. (*ibid.*) obtain a score of 97.50% using their combined approach. They further declare the problem of this TOEFL set to be "solved". All the other *WordNet*-based measures perform poorly, with accuracy not surpassing 25.0%. According to Landauer & Dumais (*ibid.*), a large sample of applicants to US colleges from non-English speaking countries took the TOEFL tests containing these items. Those people averaged 64.5%, considered an adequate score for admission to many US universities.

	RT	WN	HSO	JCN	LCH	LIN	RES	PMI-IR	LSA
TOEFL	78.75	21.88	77.91	25.00	21.88	24.06	20.31	73.75	64.38
ESL	82.00	36.00	62.00	36.00	36.00	36.00	32.66	74.00	
RDWP	74.33	23.11	45.64	22.83	23.11	22.06	21.33		

Table 3: *Percentage of correct answers for synonymy problems*

The ESL experiment (Table 3) presents similar results. Once again, the *Roget's* system is best, answering 82% of the questions correctly. The two next best systems, PMI-IR and Hirst & St-Onge fall behind, with scores of 74% and 62% respectively. All other *WordNet* measures give very poor results, not answering more than 36% of the questions. The *Roget's* similarity measure is clearly superior to the *WordNet* ones for the RDWP questions (Table 3). *Roget's* answers 74.33% of the questions, which is almost equal to a *Good* vocabulary rating according to Reader's Digest (Lewis 2000-2001), where the next best *WordNet* measure, Hirst & St-Onge, answers only 45.65% correctly. All others do not surpass 25%.

These experiments give a clear advantage to measures that can evaluate the similarity between words of different parts-of-speech.

5 Discussion

We have shown in this paper that the electronic version of the 1987 *Penguin Roget's Thesaurus* is as good as, if not better than, *WordNet* for measuring semantic similarity. The distance measure used, often called edge counting, can be calculated quickly and performs extremely well on a series of standard synonymy tests.Out of 5 experiments, *Roget's* is better than *WordNet* every time except on the Rubenstein & Goodenough list of 65 noun pairs.

The *Roget's Thesaurus* similarity measures correlate well with human judges, and perform similarly to the *WordNet*-based measures. *Roget's* shines at answering standard synonym tests.This result was expected, but remains impressive: the semantic distance measure is extremely simple and no context is taken into account, and no word sense disambiguation is performed when answering the questions. Standardized language tests appear quite helpful in evaluating of NLP systems, as they focus on specific linguistic phenomena and offer an inexpensive alternative to human evaluation.

Most of the *WordNet*-based systems perform poorly at the task of answering synonym questions. This is due in part to the fact that the similarity measures can only by calculated between nouns, because they rely on the hierarchical structure that is almost only present for nouns in *WordNet*. The systems also suffer from not being able to deal with many phrases.

The semantic similarity measures can be applied to a variety of tasks. Lexi-

cal chains (Morris & Hirst, 1991) are sequences of words in a text that represent the same topic. Links between significant words can be established using similarity measures. Many implementations of lexical chains exist, including one using our electronic *Roget's* (Jarmasz & Szpakowicz, 2003). A semantic similarity measure can be used to define the relations between words in a chain. Our lexical chain building process builds proto-chains, a set of words linked via these relations. Our implementation refines the proto-chains to obtain the final lexical chains.

Turney (2001) has used his semantic similarity metric to classify automobile and movie reviews as well as to answer analogy problems (Turney et al. 2003). In an analogy problem, the correct pair of words must be chosen amongst four pairs, for example: *cat:meow* (a) *mouse:scamper*, (b) *bird:peck*, (c) *dog:bark*, (d) *horse:groom*, (e) *lion:scratch*. To correctly answer *dog:bark*, a system must know that a *meow* is the sound that a *cat* makes and a *bark* the sound that a *dog* makes. Both of these applications can be implemented with our version of *Roget's Thesaurus*.

Acknowledgements. We thank Peter Turney, Tad Stach, Ted Pedersen and Siddharth Patwardhan for help with this research; Pearson Education for licensing to us the 1987 Penguin's *Roget's Thesaurus*.

REFERENCES

Budanitsky, Alexander & Graeme Hirst. 2001. "Semantic Distance in *WordNet*: An Experimental, Application-oriented Evaluation of Five Measures". *Proceedings of the WordNet and Other Lexical Resources Workshop at the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics* (*NAACL 2001*), 29-34. Pittsburgh, Pennsylvania.

Edmonds, Philip & Graeme Hirst. 2002. "Near-Synonymy and Lexical Choice". *Computational Linguistics* 28:2.105-144.

Fellbaum, Christiane, ed. 1998. *WordNet: An Electronic Lexical Database*. Cambridge, Mass.: MIT Press.

Hirst, Graeme & David St-Onge. 1998. "Lexical Chains as Representation of Context for the Detection and Correction of Malapropisms". *WordNet: An Electronic Lexical Database* ed. by Christiane Fellbaum, 305-332. Cambridge, Mass.: MIT Press.

Jarmasz, Mario & Stan Szpakowicz. 2001. "The Design and Implementation of an Electronic Lexical Knowledge Base". *Proceedings of the 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence* (*AI 2001*), 325-334. Ottawa, Canada.

Jarmasz, Mario & Stan Szpakowicz. 2003. "Not As Easy As It Seems: Automating the Construction of Lexical Chains Using *Roget's Thesaurus*". *16th Canadian Conference on Artificial Intelligence* (*AI 2003*), 544-549. Halifax, Canada.

Jiang, Jay J. & David W. Conrath. 1997. “Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy”. *Proceedings of the 10th International Conference: Research on Computational Linguistics* (*ROCLING X*), 19-33. Taipei, Taiwan.

Landauer, Thomas K. & Susan T. Dumais. 1997. “A Solution to Plato’s Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge”. *Psychological Review* 104:211-240.

Leacock, Claudia & Martin Chodorow. 1998. “Combining Local Context and *WordNet* Similarity for Word Sense Identification”. *WordNet: An Electronic Lexical Database* ed. by Christiane Fellbaum, 265-283. Cambridge, Mass.: MIT Press.

Lewis, Murray, ed. 2000-2001. *Reader’s Digest* 158(932, 934, 935, 936, 937, 938, 939, 940), 159(944, 948). Reader’s Digest Magazines Canada Limited.

Lin, Dekang. 1998. “An Information-Theoretic Definition of Similarity”. *Proceedings of the 15th International Conference on Machine Learning* (*ICML’98*), 296-304. Madison, Wisconsin.

McHale, Michael L. 1998. “A Comparison of *WordNet* and *Roget’s* Taxonomy for Measuring Semantic Similarity”. *Proceedings of the Workshop on Usage of WordNet in Natural Language Processing Systems at the 17th International Conference on Computational Linguistics & the 36th Annual Meeting of the Association for Computational Linguistics* (*COLING-ACL’98*), 115-120. Montréal, Canada.

Miller, George A. & Walter G. Charles. 1991. “Contextual Correlates of Semantic Similarity”. *Language and Cognitive Processes* 6:1.1-28.

Morris, Jane & Graeme Hirst. 1991. “Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text”. *Computational Linguistics* 17:1.21-48.

Pedersen, Ted. 2002. “WordNet::Similarity Perl Module”. — `www.d.umn.edu/~tpederse/similarity.html` [Source checked in May 2004].

Resnik, Philip. 1995. “Using Information Content to Evaluate Semantic Similarity”. *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (*IJCAI-95*), 448-453. Montréal, Canada.

Rubenstein, Herbert & John B. Goodenough. 1965. “Contextual Correlates of Synonymy”. *Communications of the Association for Computing Machinery* (*ACM*) 8:10.627-633.

Turney, Peter D. 2001. “Mining the Web for Synonyms: PMI-IR vs. LSA on TOEFL”. *Proceedings of the 12th European Conference on Machine Learning* (*ECML-2001*), 491-502. Freiburg, Germany.

Turney, Peter D., Michael L. Littman, Jeffrey Bigham & Victor Shnayder. 2003. “Combining Independent Modules to Solve Multiple-choice Synonym and Analogy Problems”. *Proceedings of the International Conference on Recent Advances in Natural Language Processing* (*RANLP’03*), 482-489. Borovets, Bulgaria.

Clustering WordNet Word Senses

ENEKO AGIRRE & OIER LOPEZ DE LACALLE
University of the Basque Country

Abstract

This paper presents the results of a set of methods to cluster WordNet word senses. The methods rely on different information sources: confusion matrixes from Senseval-2 Word Sense Disambiguation systems, translation similarities, hand-tagged examples of the target word senses and examples obtained automatically from the web for the target word senses. The clustering results have been evaluated using the coarse-grained word senses provided for the lexical sample in Senseval-2. We have used Cluto, a general clustering environment, in order to test different clustering algorithms. The best results are obtained for the automatically obtained examples, yielding purity values up to 84% on average over 20 nouns.

1 Introduction

WordNet (Miller et al. 1994) is one of the most widely used lexical resources for Natural Language Processing. Among other information, it provides a list of word senses for each word, and has been used in many Word Sense Disambiguation (WSD) systems as the sense inventory of choice. In particular it has been used as the sense inventory for the Senseval-2 English Lexical sample WSD exercise[1] (Edmonds & Cotton 2001).

Many works cite the fine-grainedness of the sense distinctions in WordNet as one of its main problems for practical applications (Peters et al. 1998, Tomuro 2001, Palmer et al. submitted). Senseval-2, for instance, provides both fine-grained (the actual WordNet word senses) and coarse-grained sense distinctions.

There is considerable literature on what makes word senses distinct, but there is no general consensus on which criteria should be followed. Some approaches use an abstraction hierarchy as those found in dictionaries (Kilgarriff 1998), others utilize syntactic patterns such as predicate-argument structure of verbs (Palmer et al. submitted), and others study the word senses from the point of view of systematic polysemy (Peters et al. 1998, Tomuro 2001). From a practical point of view, the need to make two senses distinct will depend on the target application.

This paper proposes a set of automatic methods to hierarchically cluster the word senses in WordNet. The clustering methods that we examine in this paper are based on the following information sources:

[1] In the rest of the paper, the Senseval-2 English lexical sample exercise will be referred to in short as Senseval-2.

(1) Similarity matrix for word senses based on the **confusion matrix** of all systems that participated in Senseval-2 (cf. Section 5).
(2) Similarity matrix for word senses produced by (Chugur & Gonzalo 2002) using **translation equivalences** in a number of languages (cf. Section 6).
(3) **Context of occurrence** for each word sense (cf. Section 7). Two methods were used to derive the contexts: taking them directly from Senseval-2 (hand tagged data), or automatically retrieving them using WordNet information to construct queries over the web (cf. Section 2).
(4) Similarity matrix based on the **Topic Signatures** for each word sense (cf. Section 8). The topic signatures were constructed based on the occurrence contexts of the word senses, which, similar to the point above, can be extracted from hand-tagged data or automatically constructed from the Web.

In order to construct the hierarchical clusters we have used Cluto (Karypis 2001), a general clustering environment that has been successfully used in Information Retrieval and Text Categorization. The input to the algorithm can be either a similarity matrix for the word senses of each target word (constructed based on confusion matrixes, translation equivalences, or topic signatures, as mentioned above), or the context of occurrence of each word sense in the form of a vector. Different weighting and clustering schemes were tried (cf. Section 4).

The paper is organized as follows. Sections 2 and 3 explain the methods to construct the corpus of word sense examples from the web and the topic signatures, respectively. Section 4 presents the clustering environment. Sections 5 through 8 present each of the methods to cluster word senses. Section 9 presents the results of the experiment. Finally, Section 11 draws the conclusions and future work. This paper is a reduced version of (Agirre & Lopez de Lacalle, 2003), where more examples and a section of related work is presented.

2 Retrieving examples for word senses from the Web

Corpora where the occurrences of word senses have been manually tagged are a scarce resource. Semcor (Miller et al. 1994) is the largest of all and currently comprises 409,990 word forms. All 190,481 open-class words in the corpus are tagged with word senses. Still, it has a low number of examples for each word sense. The word *bar*, for instance, has 6 word senses, but only 21 occurrences in Semcor.

Other tagged corpora are based on a limited sample of words. For instance, the Senseval-2 corpus comprises 5,266 hand-tagged examples for a set of 29 nouns, yielding an average of 181.3 examples per word. In particular, *bar* has 455 occurrences.

The scarcity of hand-tagged data is the acquisition bottleneck of supervised WSD systems. As an alternative, different methods to build examples for word

senses have been proposed in the literature (Leacock et al. 1998; Agirre et al. 2000, 2001). The methods usually rely on information in WordNet (lexical relations such as synonymy and hypernymy, or words in the gloss) in order to retrieve examples from large corpora or the web. The retrieved examples might not contain the target word, but they contain a word that is closely related to the target word sense.

In this work, we have followed the monosemous relatives method, as proposed in (Leacock et al. 1998). This method uses monosemous synonyms or hyponyms to construct the queries. For instance, the first sense of *channel* in Figure 1 has a monosemous synonym "*transmission channel*". All the occurrences of "*transmission channel*" in any corpus can be taken to refer to the first sense of channel. In our case we have used the following kind of relations in order to get the monosemous relatives: hypernyms, direct and indirect hyponyms, and siblings. The advantages of this method is that it is simple, it does not need error-prone analysis of the glosses and it can be used with languages where glosses are not available in their respective WordNets.

Google[2] was used to retrieve the occurrences of the monosemous relatives. In order to avoid retrieving full documents (which is time consuming) we take the context from the snippets returned by Google. Agirre et al. (2001) showed that topic signatures built from sentence context were more precise than those built from full document context.

The snippets returned by Google (up to 1,000 per query) are processed, and we try to extract sentences (or fragments of sentences) containing the search term from the snippets. The sentence (or fragment) is marked by three dots in the snippets. Some of the potential sentences are discarded, according to the following heuristics: length shorter than 6 words, the number of non-alphanumeric characters is greater than the number of words divided by two, or the number of words in uppercase is greater than those in lowercase.

3 Constructing topic signatures

Topic signatures try to associate a topical vector to each word sense. The dimensions of these topical vectors are the words in the vocabulary, and the weights try to capture which are the words closer to the target word sense. In other words, each word sense is associated with a set of related words with associated weights.

We can build such lists from a sense-tagged corpora just observing which words co-occur distinctively with each sense, or we can try to associate a number of documents from existing corpora to each sense and then analyze the occurrences of words in such documents (cf. previous section).

[2] We use the offline XML interface kindly provided by Google.

The method to construct topic signatures proceeds as follows: (i) We first organize the documents in collections, one collection per word sense, directly using sense-tagged corpora (e.g., Senseval-2), or exploiting the information in WordNet to build queries and search the web (see section 2). Either way we get one document collection per word sense. (ii) For each collection we extract the words and their frequencies, and compare them with the data in the collections pertaining to the other word senses using χ^2. (iii) The words that have a distinctive frequency for one of the collections are collected in a list, which constitutes the topic signature for the respective word sense. (iv) The topic signatures for the word senses are filtered with the cooccurrence list of the target word taken from balanced corpora such as the BNC. This last step takes out some rare and low frequency words from the topic signatures.

Topic signatures *for words* have been successfully used in summarization tasks (Lin and Hovy 2000). Agirre et al. (2000, 2001) show that it is possible to obtain good quality topic signatures *for word senses*. The topic signatures built in this work can be directly examined in `http://ixa.si.ehu.es/Ixa/resources/sensecorpus`.

4 The Cluto clustering environment

Cluto is a freely available clustering environment that has been successfully used in Information Retrieval and Text Categorization (Karypis 2001, Zhao & Karypis 2001). The input to the algorithm can be either a similarity matrix for the word senses of each target word (constructed based on confusion matrixes, translation equivalences, or topic signatures, as mentioned above), or the context of occurrence of each word sense in the form of a vector.

In the case of the similarity matrixes the default settings have been tried, as there were only limited possibilities. In the case of using directly the contexts of occurrences, different weighting schemes (plain frequencies, tf.idf), similarity functions (cosine, correlation coefficient) and clustering functions (repeated bisection, optimized repeated bisection, direct, nearest neighbor graph, agglomerative, agglomerative combined with repeated bisection) have been tried (Karypis 2001).

5 Clustering using WSD system confusion matrixes

Given the freely available output of the WSD systems that participated in Senseval-2 (Edmonds & Cotton, 2001), one can construct a confusion matrix for each pair of word senses, constructed as follows: (i) For each pair of word senses (a, b), we record the number of times that each WSD system yields a when $b$is the correct word sense in the gold standard. (ii) This number is divided by the number

channel	signals	passage	groove	water	comms	body	tv
signals	-	0.18	0.04	0.05	0.06	0.29	1.01
passage		-	0.52	0.30	0.26	0.27	1.05
groove			-	0.65	0.20	0.02	0.66
water				-	0.04	0.03	0.00
comms					-	0.45	0.64
body						-	0.55
tv							-

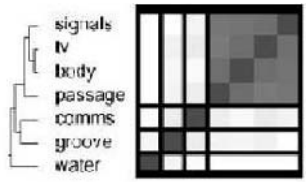

Figure 1: *Similarity matrix and resulting hierarchical clusters based on Senseval-2 topic signatures. Entropy: 0.286, Purity: 0.714*

of occurrences of word sense b and the number of systems.

The rationale is that when the systems confuse two word senses often, we can interpret that the context of the two word senses is similar. For instance, if sense a is returned instead of sense b always for all systems, the similarity of a to b will be 1. If the two senses are never confused, then their similarity would be 0. Note that the similarity matrix is not symmetric.

6 Clustering using translation similarities

Chugur & Gonzalo (2002) constructed similarity matrixes for Senseval-2 words using **translation equivalences** in 4 languages, a method proposed by Resnik & Yarowsky (2000). Two word senses are deemed similar if they are often translated with the same word in a given context. More than one language is used to cover as many word sense distinctions as possible. Chugur and Gonzalo kindly provided their similarity matrixes, and we run Cluto directly on them.

7 Clustering using word sense examples

Clustering of word senses can be cast as a document-clustering problem, provided each word sense has a pseudo-document associated with it. This pseudo-document is built combining all occurrences of the target word sense (e.g., following the methods in Section 2). Once we have such a pseudo-document for each word sense, we can cluster those documents with the usual techniques, and the output will be clusters of the word senses.

We have used two sources to build the pseudo-documents. On the one hand we have used Senseval 2, using the sentence context only. On the other hand we have used the examples collected from the web, as explained in Section 2. Table 1 shows, among other information, the number of examples available for each of the words considered. A range of clustering parameters was tried on these sets of examples, as explained on Section 4.

noun	#s	#c	#sval	#web	pur	noun	#s	#c	#sval	#web	pur
art	4	2	275	23391	0.750	fatigue	4	3	116	8596	1.000
authority	7	4	262	108560	0.571	feeling	6	4	153	14569	1.000
bar	13	10	360	75792	0.769	hearth	3	2	93	10813	0.667
bum	4	3	115	25655	1.000	mouth	8	5	171	1585	0.833
chair	4	2	201	38057	0.750	nation	4	2	101	1149	1.000
channel	7	4	181	46493	0.714	nature	5	3	137	44242	0.600
child	4	2	189	70416	0.750	post	8	5	201	55049	0.625
circuit	6	4	247	33754	0.833	restraint	6	4	134	49905	0.667
day	9	5	427	223899	1.000	sense	5	4	158	13688	0.800
facility	5	2	172	17878	1.000	stress	5	3	112	14528	0.800

Table 1: *List of nouns processed in this study. The columns show the number of senses (#s), number of clusters in the gold standard (#c), number of examples in Senseval-2 (#sval), number of examples retrieved from the web (#web), and best purity value obtained for each word (pur)*

8 Clustering using topic signatures

Topic signatures have been constructed for the Senseval 2 nouns using two sets of examples as in the previous section: the Senseval 2 hand-tagged data, and the automatically retrieved sense examples. Once the topic signatures were constructed the similarity matrix was built using the cosine as similarity function among topic signatures. Figure 1 shows the similarity matrix for channel based on topic signatures, and the resulting hierarchical cluster.

9 Experiments

In order to evaluate the clustering results we have used as reference gold standard the coarse senses for nouns provided by Senseval-2. This gold standard is used in order to compute purity and entropy values for the clustering results (see below). The number of resulting groups is used as the target number of clusters. In the case of authority (cf. Table 1) there are 4 sense groups in the gold standard, and therefore Cluto is instructed to build 4 clusters.

Some of the nouns in Senseval-2 had trivial clustering solutions, e.g., when all the word senses form a single cluster, or all clusters are formed by a single word sense. Table 1 shows the 20 nouns that had non-trivial clusters and could therefore be used for evaluation.

The quality of a clustering solution was measured using two different metrics that looked at the gold-standard labels of the word senses assigned to each cluster (Zhao & Karypis 2001). In order to better explain the metrics, we will call the gold-standard sense groups *classes*, as opposed to the clusters returned by the

Method	Entropy	Purity
Random	-	0.748
Confusion Matrixes	0.364	0.768
Multilingual Similarity	0.337	0.799
Examples: Senseval (Worse)	0.378	0.744
(Best)	0.338	0.775
Examples: Web (Worse)	0.310	0.800
(Best)	**0.209**	**0.840**
Topic Signature: Senseval	0.286	0.806
Topic Signature: Web	0.358	0.764

Table 2: *Entropy and purity results for the different clustering methods*

methods. The first metric is the widely used *entropy* measure that looks are how the various classes of word senses are distributed within each cluster, and the second measure is the *purity* that measures the extent to which each cluster contained word senses from primarily one class. A perfect clustering solution will be the one that leads to clusters that contain word senses from only a single class, in which case the entropy will be zero. In general, the smaller the entropy values, the better the clustering solution is. The purity of one cluster is defined to be the fraction of the overall cluster size that the largest class of word senses assigned to that cluster represents. The overall purity of the clustering solution is obtained as a weighted sum of the individual cluster purities. In general, the larger the values of purity, the better the clustering solution is. Evaluation of clustering solutions is not easy, and usually both measures are provided.

As a baseline we built a random baseline, which was computed averaging among 100 random clustering solutions for each word. Each clustering solution was built assigning a cluster to each word sense at random.

Table 2 shows the results in the form of average entropy and purity for each of the clustering methods used. The first line shows the results for the random baseline. For the confusion matrix and multilingual matrix a single solution is shown. In the case of clustering directly over the examples, the best and worst clustering results are shown for all the combinations of parameters that were tried. Space limits prevent us from showing all combinations. Different results are shown depending on whether the examples were taken from Senseval-2 or from the web examples. Finally, the results over the topic signatures are shown, with different lines depending on the source of the examples.

According to the results, automatically obtained examples proved the best : the optimal combination of clustering parameters gets the best results, and the worst parameter setting gets the third best results. Thus, automatically retrieved examples from the Web are the best source for replicating the gold standard from the alternatives studied in this paper.

If we compare the Web results with the Senseval-2 results (which according to the quality of the hand-tagged data should be better) we see that direct clustering on the examples yields very bad results for Senseval-2 (below random for worst parameter setting). Topic signatures on Senseval-2 data, on the contrary, provide the second best results. We think that the main difference between Senseval-2 and Web data is the amount of data (cf. table 1). It seems that topic signatures provide useful clusters when little data is available, while direct clustering is best for large amounts of data.

We want to note that the random baseline is quite high. Still, the reduction of error for the best result (measured according to purity) is of 35%, i.e., error is reduced from 0.252 to 0.16. We want to note that related literature seldom cite random baselines for clustering problems.

10 Conclusions and future work

This paper has presented the results of a set of methods to cluster WordNet word senses. The methods rely on different information sources: confusion matrixes from Senseval-2 systems, translation similarities, hand-tagged examples of the target word senses and examples obtained automatically from the web for the target word senses. The clustering results have been evaluated using the coarse-grained word senses provided for the lexical sample in Senseval-2. We have used Cluto, a general clustering environment, in order to test different clustering algorithms. An extended version of this paper (Agirre & Lopez de Lacalle, 2003) shows more examples, and a full section on related work.

The best results are obtained with the automatically obtained examples, with purity values up to 84% on average over 20 nouns.

We are currently acquiring 1,000 snippets from Google for each monosemous noun in WordNet. The total amount of monosemous nouns in version 1.6 is 90,645 and we have currently acquired examples for 98% of them. The examples take nearly 16 Gigabytes. We plan to provide word sense clusters of comparable quality for all nouns in WordNet soon[3].

We also plan to perform a task based evaluation, using the hierarchical clusters to guide a WSD algorithm, and to further investigate the relation of the produced clusters and studies of systematic polysemy.

Acknowledgements. We want to thank Irina Chugur and Julio Gonzalo for kindly providing us with the multilingual data. This work has been funded by the European Commision (project MEANING IST-2001-34460) and the MCYT (project HERMES 2000-0335-C03-03).

[3] Check `http://ixa.si.ehu.es/Ixa/resources/sensecorpus`

REFERENCES

Agirre, Eneko, Olatz Ansa, David Martinez & Eduard Hovy. 2000. "Enriching Very Large Ontologies with Topic Signatures". *Proceedings of the workshop on "Ontology Learning", held in conjunction with the European Conference on Artificial Intelligence* (*ECAI*), 25-30. Berlin, Germany.

Agirre, Eneko, Olatz Ansa, David Martinez & Eduard Hovy. 2001. "Enriching WordNet Concepts with Topic Signatures". *Proceedings of the SIGLEX Workshop on "WordNet and Other Lexical Resources: Applications, Extensions and Customizations" held in conjunction with the meeting of the North American chapter of the Association of Computational Linguistics*, 23-28. Pittsburgh, Pennsylvania.

Agirre, Eneko & Oier Lopez de Lacalle. 2003. "Clustering WordNet Word Senses". *Proceedings of the Conference on Recent Advances on Natural Language Processing* (*RANLP'03*), 11-18. Borovets, Bulgaria. Also available at `http://ixa.si.ehu.es/Ixa/Argitalpenak/Artikuluak/1068721391/publikoak/agirre.pdf` [Source checked in May 2004]

Chugur, Irina & Julio Gonzalo. 2002. "A study of Polysemy and Sense Proximity in the Senseval-2 Test Suite". *Proceedings of the Association of Computational Linguistics Workshop on "Word Sense Disambiguation: Recent Successes and Future Directions"*, 32-39. Philadelphia, Pennsylvania.

Edmonds, Phil & Scott Cotton, eds. 2001. *Proceedings of the Senseval-2 Workshop, held in conjunction with the annual meeting of the Association for Computational Linguistics*. Toulousse, France.

Karypis, George. 2001. "CLUTO. A Clustering Toolkit". Technical Report (#02-01). Minneapolis, U.S.A.: Dept. of Computer Science, Univ. of Minnesota. Also available at `http://www.cs.umn.edu/~karypis`. — [Source checked in May 2004]

Kilgarriff, Adam. 1999. "Inter-Tagger Agreement". *Advanced papers of the SENSEVAL Workshop*. Brighton, U.K. Also available at `http://www.itri.brighton.ac.uk/events/senseval/ARCHIVE/PROCEEDINGS/`. — [Source checked in May 2004]

Leacock, Claudia, Martin Chodorow & George A. Miller. 1998. "Using Corpus Statistics and WordNet Relations for Sense Identication". *Computational Linguistics* 24:1.147-166.

Lin, Chin-Yew & Eduard Hovy. 2000. "The Automated Acquisition of Topic Signatures for Text Summarization". *Proceedings of the Computational Linguistics Conference* (*COLING-2000*), 495-501. Strasbourg, France.

Miller, George A., Richard Beckwith, Christiane Fellbaum, David Gross & K. Miller. 1990. "Five Papers on WordNet". *Special Issue of the International Journal of Lexicography* 3:4.235-264.

Palmer, Marta, Hoa T. Trang & Christiane Fellbaum. Forcoming. "Making Fine-Grained and Coarse-Grained Sense Distinctions, both Manually and Automatically". To appear in *Natural Language Engineering*.

Resnik, Philip & David Yarowsky. 2000. “Distinguishing Systems and Distinguishing Senses: New Evaluation Methods for Word Sense Disambiguation”. *Natural Language Engineering* 5:2.113-133.

Tomuro, Noriko. 2001. “Tree-cut and A Lexicon Based on Systematic Polysemy”. *Proceedings of the 2nd Meeting of the North American Association for Computational Linguistics*, 71-78. Pittsburgh, Pennsylvania.

Zhao, Ying & George Karypis. 2001. “Criterion Functions for Document Clustering: Experiments and Analysis”. Technical Report (TR#01–40). Minneapolis, U.S.A.: Dept. of Computer Science, Univ. of Minnesota.

Inducing Hyperlinking Rules in Text Collections

ROBERTO BASILI, MARIA TERESA PAZIENZA &
FABIO MASSIMO ZANZOTTO
University of Rome "Tor Vergata"

Abstract

Defining link types and writing linking rules for automatic hyperlinking methods may be a cumbersome activity due to the large number of possibilities. In this paper, we tackle this issue proposing a model for automatically extracting link types and, as a consequence, linking rules from large text collections. The novel idea is to exploit relations among facts expressed within domain documents.

1 Introduction

Hyperlinked text collections are often seen as an added value. On-line news agencies, for instance, tend to offer this service to their customers. But, tracing hyperlinks between documents is an inherently difficult task. As the process of writing, it involves the ability of finding relations among concepts or facts. It is not surprising that the inter-agreement among annotators is very low even if they are only asked to produce links between "related texts" (see Ellis et al. 1994). The disagreement is even bigger if the tagert is to relate documents with typed links. Link types as "cause-effect" relations may help in filtering out irrelevant information as final users may better decide if it is worthy or not to traverse provided links.

In Basili et al. (2003), we proposed a hyper-linking method based on Information Extraction techniques conceived to connect documents with typed relations. Linking among documents is based on an intermediate representation, called the "objective representation". This document surrogate contains only events judged relevant according to an underlying domain knowledge-base. On the basis of rules firing on the event classes, a link is justified according to events (and the involved entities) appearing in the two documents. The model offers a language over which specific linking rules may be manually written building on the supported event classes. However, the activity of defining link types and writing the related linking rules may not be an easy task mainly when a large number of fact classes is foreseen.

In this paper, we want to tackle this last issue by proposing a model for the automatic definition of link types and the related linking rules in the context of a rule-based hyper-linking method. The basic assumption we make is the one we started with: *the activity of building hypertexts is very similar to the process of writing*. Therefore, the novel idea is to exploit the relations among facts as they

appear inside domain documents for inducing hyperlinking rules. In our opinion, typical discourse structures of domain documents are valuable resources for defining relevant relation types and the related linking rules. Trusting on such an assumption we propose a light discourse model able to infer regularities in documents belonging to a collection. Co-occurrences of event classes will be used to derive link types and linking rules. As the exploration of the proposed model demands for a linguistic analysis of texts applied over large amount of data, we refer to assessed language processing technologies. Therefore, we build on robust methods for processing natural language (i.e., a robust syntactic parser (Basili & Zanzotto 2002) and a shallow knowledge-based method for a semantic analysis such as in Humphreys et al. (1998)) and on well assessed statistical measures (i.e., the mutual information). It is worth noticing that insights borrowed from the Discourse Analysis theory have also fashioned the definition of "generic" link types such as *revision*, *association*, and *aggregation* (Parunak 1991).

To better explain our model for deriving hyper-linking rules, we will firstly summarize the rule-based approach to hyper-linking (Section 2). Then we will both introduce our light model for inspecting the discourse (based on a robust syntactic parser and a knowledge-based shallow semantic analyser) and describe how the proposed model enables the extraction of the relevant and stable relations among event types using statistical measures (Section 3). Finally, the viability of the approach is inspected in the financial domain represented by a document collection of news articles (Section 4).

2 Rule-based hyperlinkingwith information extraction techniques

The identification of links among texts in document collections may be a very subjective matter according to the perception of what is useful. Therefore, a hypertext should take into account that the final users may ask very different supports in reasoning. As Glushko (1989) observers, some hypertextual links may even be misleading for a user not interested in the information they are pointing to. Assigning types to hyper-links is then suggested to overcome such a problem. From the point of view of the possible services, a typed text network is useful for supporting a personalised access to the linking information according to dinamically changing specific needs. The extent of the personalisation depends on the used link type system T and on the availability of automatic methods able to assign types in T to the retrieved links.

In this context, computational approaches based on bag-of-word models (e.g., (Allan 1996)) or on more semantic document representations such as *lexical chains* (Hirst 1991) used in Green (1997) may not be very well suited. These models can offer a degree of relatedness among documents that is not sufficient to classify links in types as "cause-effect". In Allan (1996) this type is consid-

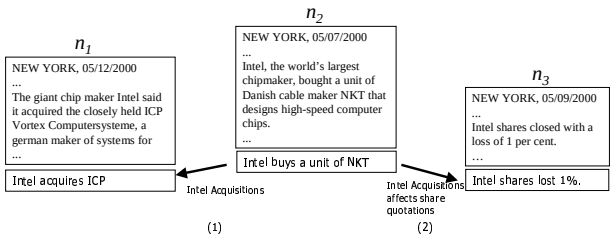

Figure 1: *Examples of justified links*

ered "manual", i.e., *out of scope* for these computational models. Detectable link types are in 6 classes called "automatic" links: *revision*, *summary/expansion*, *equivalence*, *comparison/contrast*, *tangent*, and *aggregate*. To push forward the limit, deeper text understanding models are needed. For example, while these approaches may relate the news items n_1, n_2, and n_3 in Figure 1 using the `Intel` stem, they cannot produce any further linking motivation. On the other hand, it may be relevant to make explicit that the link between the two news items n_1 and n_2 is motivated by the fact that both deal with the `Intel acquisition activity` or that the link between n_2 and n_3 describes a kind of cause-effect.

In the approach we follow (Basili et al. 2003), we rely on an *objective representation* (OR) of the document content to support the automatic extraction of typed links among documents. A document d is represented as a sequence of typed events, i.e.,

$$d = \{e_1, ..., e_n\} \tag{1}$$

where each event $e_i = (t_i, \{a_i^1, a_i^2, ..., a_i^n\})$ is represented by a type $t(e_i) = et_i$ and a set of arguments $p(e_i) = \{a_i^1, a_i^2, ..., a_i^n\}$ related to the participants. For instance, given the event class buy_event and $share\_trend$, the three news articles in Figure 1 would have respectively the following *objective representations*:

$$\begin{array}{l} \overline{n}_1 = \{(buy\_event, \{agent(Intel), patient(ICP)\})\}, \\ \overline{n}_2 = \{(buy\_event, \{agent(Intel), patient(a\_unit\_of\_NKT)\})\}, and \\ \overline{n}_3 = \{(share\_trend, \{agent(Intel), modifier(1\%)\})\}. \end{array} \tag{2}$$

On this document model, a powerful language for the description of complex linking rules has been provided (see (Basili et al. 2003) for more details). Hyperlinking rules may be written as logical forms in which triggers justify firings. These rules state that two documents d_1 and d_2 are related with a link type $lt(et_1, et_2)$ if (i) an event of the type et_1 is detected in d_1 and one of type et_2 in d_2, and (ii) the two events satisfy some equivalence constraints on the partecipants. In this language cause-effect linking rules may be easily written. For example, relevant and completely defined link types able to explain, respectively,

the relations (1) and (2) in Figure 1 are $lt(buy\_event, buy\_event)$ (a *same event* link type) and $lt(buy\_event, share\_trend)$ (a sort of *cause-effect*). Link types, $lt(et_1, et_2)$, and linking rules are strictly correlated: both are completely defined when the types of the involved events have been identified.

The rule-based hyper-linkingparadigm may support the automatic construction of a typed text networks. Different types of links may be defined according to the foreseen information needs. However, the definition of the hyper-linking types, i.e., $lt(et_1, et_2)$, and the related rules may still be a cumbersome problem since prototypical relations among facts may not be so evident. The problem is strictly related to the number of the possible event classes (the set EC) the underlying information extraction system is able to deal with. A large EC set results in a rich language for expressing the link types and very specific rules may be foreseen. But, as the size of EC grows, the hyperlinking rule definition activity becomes more complex as the number of possible linking types to be considered grows squarely. Trivially, the number of types $lt(et_1, et_2)$ with $et_1, et_2 \in EC$, whose relevance has to be judged, is $|EC|^2/2$. Aiming to reduce this space, we here propose a method to suggest (among all the possible ones) the more promising link types among event classes that may be easily translated in hyper-linking rules.

3 Shallow discourse analysis for automatically deriving hyperlinking rules

Establishing relations among facts and ideas is a relevant cognitive process. It supports not only the activity of tracing hyper-links among different texts but also the writing process. We argue that if a type of relation among events is relevant in a particular domain it has been expressed in some of the texts belonging to the analysed collection. An analysis of stable relations among facts within the texts may suggest the relvant types of hyper-links and, consequently, the hyperlinking rules. In this perspective, the relations expressed in the corpus are retained as relevant link types.

Then, the model we here propose will analyse the content of the texts in light of the event classes EC and will try to find these correlations (i.e., the co-occurrences) among elements in EC that are more promising in the corpus C. The experimental framework will build on robust methods for processing natural language (i.e., a robust syntactic parser (Basili & Zanzotto 2002) and a shallow knowledge-based method for a semantic analysis in line with approaches such as (Humphreys et al. 1998), and on well assessed statistical measures (i.e., the mutual information). In the model we propose, the syntactic analyser extracting the grammatical structures of a given text cooperates with the semantic analyser to map different surface representations to the implied event class. The corpus C, reduced to a set of forms as (1), will be used to define the statistical model able to capture the stable relations among fact types by means of the mutual information scores.

3.1 *Robust syntactic parsing and knowledge-based semantic analysis*

As we need the *objective representation* to capture relations among fact classes, salient events expressed in the documents need to be made explicit. The recognition process starts from a grammatical analysis to filter out grammatical differences of surface forms and to make evident syntactic relations among the textual fragments. We will rely on a robust approach that produces partial and possibly ambiguous syntactic analysis in extended dependency graph (XDG) (for more details refer to Basili & Zanzotto (2002)). Verbal phrases, relevant for our analysis, can be straightforwardly extracted selecting the subgraphs with all the constituents that can be directly reached by a verb.

The syntactic processor SP is then a function mapping documents in raw texts t to syntactically analysed documents sd, i.e., $SP(t) = sd$. The syntactic representation sd of a document t is a set of verb contexts c, i.e., $sd = \{c_1, ..., c_n\}$. For instance, the syntactic representation for the documents n_1 and n_2 in Figure 1 are $SP(n_1) = \{c_1\}$ and $SP(n_2) = \{c_2\}$ where $c_1 = (buy, \{lsubj(company(Intel), lobj(unit\_of(company(NKT))\})$ and $c_2 = (acquire, \{lsubj(company(Intel), lobj(company(ICP))\})$.

Syntactically processed documents are still not completely generalised to allow the analysis of the co-occurrences for two given event types. Differences between surface forms still exist, e.g., no explicit equivalence is stated between the verb context c_1 and c_2. In line with Humphreys et al. (1998), we will perform this harmonisation in the semantic analysis by matching the current verb contexts with event prototypes contained in a domain knowledge base KB. This latter stores the different surface representations associated to each event type postulated in the domain (i.e., for each element in the set EC).

For the shallow semantic analysis we propose a function S that maps documents in the syntactic space to their semantic representation, i.e., $d = S(sd)$ where sd is a document syntactically represented while d is the document seen as a sequence of salient facts. The key element of the semantic analysis is the event knowledge-base KB that contains elements, *event prototypes*:

$$(t, v, [a_1, ..., a_n]) \tag{3}$$

describing that a possible form of an event typed $t \in EC$ is a verb context governed by the verb v with a_1, ..., a_n as arguments. An instance of event prototype of the type buy_event may be:

$$(buy\_event, acquire, \{lsubj(company(X)), lobj(company(Y))\}) \tag{4}$$

that is a buy_event may be represented in a context of the verb $acquire$ if the logical subject X (referred as $lsubj$) is a $company$ and the logical object Y (referred as $lobj$) is $company$. The knowledge-base KB is then used to define

the function $S(sd) = \{s(c_1), ..., s(c_n)\}$ where:

$$s(c) = \begin{cases} (t, args) & if \quad \exists (t, v, args) \in best\_match(c, KB) \\ none & otherwise \end{cases} \tag{5}$$

That is, a verbal context $c = (v, sargs)$ is transformed into an event $(t, args)$ of the type t if an event prototype $(t, v, args)$ is foreseen in the knowledge base KB whose arguments $args$ are matched against the arguments of the actual instance $sagrs$. If more than one prototype satisfies the constraints, the one matching the largest number of arguments is selected, i.e., $best\_match(c, KB)$. The default value $none$ is assigned when the verb context does not match any event prototype.

3.2 *Highlighting stable relations among fact types using mutual information*

As the final step of our model, we propose here to study the correlation among event classes using the mutual information on a statistical model of the corpus. Pairs of event classes found with a positive level of mutual information will be retained as useful to build hyper-linking rules. We assume that the link type is defined by the pair of correlated events.

In order to evaluate the mutual information (MI), two sample spaces S_1 and S_2 have been defined on the analysed corpus C. S_1 is the space of all the events occurred in the corpus C while S_2 represents all the events that jointly appeared in one of the documents. The probability that the single event e is of type t can be estimated using the frequency of the events of type t in the space S_1. Similarly, the probabilities of (et_1, et_2) pairs can be estimated on the space S_2. Then, the mutual information $mi(et_1, et_2)$ of two event types et_1 and et_2 belonging to the event classes foreseen in the model of the knowledge domain is:

$$mi(et_1, et_2) = log \frac{p(et_1, et_2)}{p(et_1)p(et_2)} \tag{6}$$

The mutual information in the corpus C of the pairs (et_1, et_2) makes evident the relations considered relevant by the writers of the analysed documents. Higher is the value of the mutual information wider is the perception of a relevant link between the two event classes. Event type pairs with a positive value of mutual information can be retained as relevant link type $lt(et_1, et_2)$ and easly translated to hyper-linking rule. Furthermore, in a supervised environment, the use of the mutual information as index offers the possibility to rank the choices according to their relevance.

1	RELATIONSHIPS AMONGS COMPANIES
1-1	Acquisition/Selling
1-2	Cooperation/Splitting
2	INDUSTRIAL ACTIVITIES
2-1	Funding/Capital
2-2	Company Assets (Financial Performances , Balances, Sheet Analysis)
2-3	Market Strategies and plans
2-4	Staff Movement (e.g., Management Succession)
2-5	External Communications
3	GOVERNMENT ACTIVITIES
3-1	Tax Reduction / Increase
3-2	Anti-Trust Control
4	JOB MARKET - MASS EMPLOYMENT/UNEMPLOYMENT
5	COMPANY POSITIONING
5-1	Position vs Competitors
5-2	Market Sector
5-3	Market Strategies and plans
6	STOCK MARKET
6-1	Share Trends
6-2	Currency Trends

Table 1: *The event class hierarchy of the financial domain*

4 Analysis of the results in a financial domain

Our model has been applied to a financial domain and the experiment has been carried out on a large document collection with these parameters: (1) the set of event classes EC and (2) the semantic knowledge base KB. We will hereafter refer to $EC + KB$ as the model for the domain. The event classes foreseen for this domain are shown in Table 1 structured into an hierarchy. These 20 classes are the same proposed in Basili et al. (2002) to describe the relevant events that may occur in a financial news stream. This class hierarchy covers event types involving the activities of the companies in the market (e.g., the class 5) as well as the events that may happen in the stock exchange (i.e., the types 6, 6-1, and 6-2). On the other hand, the knowledge base KB including about 500 event prototypes has been extracted from the corpus with the semi-automatic extraction method based on a terminological perspective (Basili et al. 2002). The terminological model enables the extraction of event prototypes from the corpus and their ranking according to a relevance score. The ranked prototypes are then shown to one or more annotators that provide a classification in one of the foreseen classes in EC. It is worth noticing that also the modular syntactic parser we are using depends itself on a number of parameters, i.e., the rules and

the lexicons it uses. In particular, due to the nature of the information we want to gather, the main parameters are the part-of-speech tagger and the verb argument detector. Nevertheless, even with all these inherent limitations we carried out the experiments on a large corpus, i.e., a collection of about 12,300 news items of the Finacial Times published in the period Oct–Dec 2000. This collection has been assumed to be the implicit domain model where to extract the domain "linking" knowledge.

After the syntactic/semantic analysis it emerges that an average of 0.89 events for each document have been matched and this can drastically reduce the expectation of finding correlations among event types. Focussing on the 5,723 out of the 12,308 documents that had a least one detected event, the average number of events per document is more that doubled (i.e., 1.91 events/docs). This value suggests that when the document is covered by the $EC + KB$ model the correlation between fact types in EC is a significant phenomenon and, consequently, can be studied. The mutual information among elements in EC is depicted in Table 2

	Hyper-link prototype		MI
lt(6-1,6)	Share Trends	STOCK MARKET	1.0574
lt(3,5-2)	GOVERNMENT ACTIVITIES	Market Sector	0.9879
lt(2-2,6-1)	Company Assets	Share Trends	0.9513
lt(2-4,2-4)	Staff Movement	Staff Movement	0.8176
lt(2-2,6)	Company Assets	STOCK MARKET	0.7785
lt(6-1,6-1)	Share Trends	Share Trends	0.5884
lt(5-2,5-2)	Market Sector	Market Sector	0.5499
lt(6,6)	STOCK MARKET	STOCK MARKET	0.3810
lt(1-2,1-2)	Cooperation/Splitting	Cooperation/Splitting	0.3462
lt(1-2,4)	Cooperation/Splitting	JOB MARKET	0.3090
lt(2-2,2-2)	Company Assets	Company Assets	0.2793
lt(1-1,1-2)	Acquisition/Selling	Cooperation/Splitting	0.1724
lt(2-4,5-1)	Staff Movement	Position vs Competitors	0.1516
lt(2-5,6-1)	External Communications	Share Trends	0.0808
lt(2-1,2-2)	Funding/Capital	Company Assets	0.0363
lt(1-1,1-1)	Acquisition/Selling	Acquisition/Selling	0.0254
lt(6,5-2)	STOCK MARKET	Market Sector	0.0210

Table 2: *Hyperlinking Rules derived from the Economic Corpus*

where hyper-linking rules among the types t_1 and t_2 are referred as $lt(t_1, t_2)$. As expected only a small part (17 out from 210) are selected and therefore presented as useful hyper-linking rules. In such a way the 90% of the possible hyper-link types (and related rules) are removed. This demonstrates that our method is a viable solution for discovering relevant link types. An evaluation of the pairs with respect to human judgements is difficult because two event types randomly picked from EC seem to be correlated. We can only discuss the obtained list

with respect to the document collection. We may observe that some event types tend to occur with events of the same type as, for instance, 2-4, 6-1, 1-1, and 2-2. This interesting fact confirms that rules connecting events of the same type can be relevant for a perspective user of the linking information. The most relevant suggested relations are those connecting different event classes. Apart from the first between a class 6-1 with its superclass 6, these relations suggest that cause-effect relation types are active in the corpus according to the domain model given by EC and, therefore, the related linking rules may be foreseen. The secondly ranked, for instance, states that a relevant relation may be drawn between the *government activities* (class 3) and the trends in a *market sector* (class 5-2). The third suggested rule states a relation between the *Company Assets* (2-2) and the *Share Trends* (6-1) similarly stated by the relation (2-2,6).

5 Conclusions

Building on the basic idea that relevant link types are already expressed in the collections of domain documents, we presented a novel method for automatically deriving hyper-linking types $lt(et_1, et_2)$ and, consequently, hyper-linking rules. The proposed method based on natural language processing techniques seems to be a viable solution to address the definition of such types and rules: in fact, when documents are covered by the domain knowledge model, the stated relations among event types may be recovered. The method we propose reduces the linking rules that have to be considered.

REFERENCES

Allan, James. 1996. “Automatic Hypertext Link Typing”. *Proceedings of the 7th ACM Conference on Hypertext*, 42-52. Washington, D.C.: ACM Press.

Basili, Roberto & Fabio Massimo Zanzotto. 2002. “Parsing Engineering and Empirical Robustness”. *Natural Language Engineering* 8:2-3.97-120.

Basili, Roberto, Alessandro Moschitti, Maria Teresa Pazienza & Fabio Massimo Zanzotto. 2003. “Personalizing Web Publishing via Information Extraction”. *IEEE Intelligent Systems* 18:1.62-70.

Basili, Roberto, Maria Teresa Pazienza & Fabio Massimo Zanzotto. 2002. “Learning IE Patterns: A Terminology Extraction Perspective”. *Workshop of Event Modelling for Multilingual Document Linking*, 36-41. Las Palmas, Spain.

Ellis, David, Jonathan Furner-Hines & Peter Willett. 1994. “The Creation of Hypertext Linkages in Full-text Documents: Parts I and II”. Technical Report RDD/G/142. London: British Library Research and Development Dept.

Glushko, Robert J. 1989. “Design Issues for Multi-Document Hypertexts”. *Proceedings of the 2nd Annual ACM Conference on Hypertext*, 51-60. Pittsburgh, Pennsylvania: ACM Press.

Green, Stephen. 1997. "Automatically Generating Hypertext by Computing Semantic Similarity". PhD dissertation. Dept. of Computer Science, Univ. of Toronto.

Humphreys, Kevin, Robert Gaizauskas, Saliha Azzam, Chris Huyck, Brian Mitchell, Hamish Cunningham & Yorick Wilks. 1998. "University of Sheffield: Description of the LASIE-II System as Used for MUC-7". *Proceedings of the 7th Message Understanding Conferences* (*MUC-7*), 84-89. San Fransisco, Calif.: Morgan Kaufmann.

Morris, Jane & Graeme Hirst. 1991. "Lexical Cohesion, the Thesaurus, and the Structure of Text". *Computational Linguistics* 17:1.21-48.

Van Dyke Parunak, Henry. 1991. "Ordering the Information Graph". *Hypertext/ Hypermedia Handbook* ed. by Emily Berk & Joseph Devlin, 299-325 (= chapter 20). Hightstown, N.J.: McGraw-Hill (Intertext).

Near-Synonym Choice in Natural Language Generation

DIANA ZAIU INKPEN* & GRAEME HIRST**

*University of Ottawa, School of Information Technology and Engineering
**University of Toronto, Department of Computer Science

Abstract

We present Xenon, a natural language generation system capable of distinguishing between near-synonyms. It integrates a near-synonym choice module with an existing sentence realization module. We evaluate Xenon using English and French near-synonyms.

1 Introduction

Natural language generation systems need to choose between *near-synonyms* — words that have the same meaning, but differ in lexical nuances. Choosing the wrong word can convey unwanted connotations, implications, or attitudes. The choice between near-synonyms such as *error*, *mistake*, *blunder*, and *slip* can be made only if knowledge about their differences is available.

In previous work (Inkpen & Hirst 2001) we automatically built a lexical knowledge base of near-synonym differences (LKB of NS). The main source of knowledge was a special dictionary of near-synonym discrimination, *Choose the Right Word* (Hayakawa 1994). The LKB of NS was later enriched (Inkpen 2003) with information extracted from other machine-readable dictionaries, especially the Macquarie dictionary.

In this paper we describe Xenon, a natural language generation system that uses the knowledge of near-synonyms. Xenon integrates a new near-synonym choice module with the sentence realization system named HALogen (Langkilde & Knight 1998, Langkilde-Geary 2002). HALogen is a broad-coverage general-purpose natural language sentence generation system that combines symbolic rules with linguistic information gathered statistically from large text corpora (stored in a language model). For a given input, it generates all the possible English sentences and it ranks them according to the language model, in order to choose the most likely sentence as output.

Figure 1 presents Xenon's architecture. The input is a semantic representation and a set of preferences to be satisfied. A concrete example of input is shown in Figure 2. The final output is a set of sentences and their scores. The highest-ranked sentence is considered to be the solution.

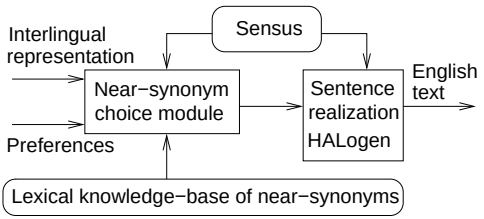

Figure 1: *The architecture of Xenon*

2 Meta-concepts

The semantic representation that is one of Xenon's inputs is represented, like the input to HALogen, in an interlingua developed at ISI (Information Science Institute, University of Southern California). This language contains a specified set of 40 roles, and the fillers of the roles can be words, concepts from Sensus (Knight & Luk 1994), or complex representations.

Xenon extends the representation language by adding meta-concepts. The meta-concepts correspond to the core denotation of the clusters of near-synonyms, which is a disjunction (an OR) of all the senses of the near-synonyms of the cluster.

We use distinct names for the various meta-concepts. The name of a meta-concept is formed by the prefix "generic", followed by the name of the first near-synonym in the cluster, followed by the part-of-speech. For example, if the cluster is *lie, falsehood, fib, prevarication, rationalization, untruth*, the name of the cluster is "generic_lie_n". If there are cases where there could still be conflicts after differentiating by part-of-speech, the name of the second near-synonym is used. For example, "stop" is the name of two verb clusters; therefore, the two clusters are renamed: "generic_stop_arrest_v" and "generic_stop_cease_v".

3 Near-synonym choice

Near-synonym choice involves two steps: expanding the meta-concepts, and choosing the best near-synonym for each cluster according to the preferences.

Input: (A9 / tell :agent (V9 / boy) :object (O9 / generic_lie_n)

Input preferences: ((DISFAVOUR :AGENT)
(LOW FORMALITY) (DENOTE (C1 / TRIVIAL)))

Output: The boy told fibs. −40.8177

Figure 2: *Example of input and output of Xenon*

```
(A9 / tell    :agent (V9 / boy)
  :object (OR
     (e1 / (:CAT NN :LEX "lie") :WEIGHT 1.0e-30)
     (e2 / (:CAT NN :LEX "falsehood") :WEIGHT 6.93e-8)
     (e3 / (:CAT NN :LEX "fib") :WEIGHT 1.0)
     (e4 / (:CAT NN :LEX "prevarication") :WEIGHT 1e-30)
     (e5 / (:CAT NN :LEX "rationalization") :WEIGHT 1e-30)
     (e6 / (:CAT NN :LEX "untruth") :WEIGHT 1.38e-7))
```

Figure 3: *The interlingual representation from Fig. 2 after expansion by the near-synonym choice module.*

We implemented this in a straightforward way: the near-synonym choice module computes a satisfaction score for each near-synonym; then satisfaction scores become weights; in the end, HALogen makes the final choice, by combining these weights with the probabilities from its language model. For the example in Figure 2, the expanded representation, which is input to HALogen, is presented in Figure 3. The near-synonym choice module gives higher weight to *fib* because it satisfies the preferences better than the other near-synonyms in the same cluster. Section 4 will explain the algorithm for computing the weights.

4 Preferences

The preferences that are input to Xenon could be given by the user, or they could come from an analysis module if Xenon is used in a machine translation system (corresponding to nuances of near-synonyms in a different language). The formalism for expressing preferences and the preference satisfaction mechanism is adapted from the prototype system I-Saurus (Edmonds & Hirst 2002).

The preferences, as well as the distinctions between near-synonyms stored in the LKB of NS, are of three types. Stylistic preferences express a certain formality, force, or concreteness level and have the form: (strength stylistic-feature), for example (low formality). Attitudinal preferences express a favorable, neutral, or pejorative attitude and have the form: (stance entity), where stance takes the values favour, neutral, disfavour. An example is: (disfavour :agent). Denotational preferences connote a particular concept or configuration of concepts and have the form: (indirectness peripheral-concept), where indirectness takes the values suggest, imply, denote. An example is: (imply (C / assessment :MOD (OR ignorant uninformed))). The peripheral concepts are expressed in the ISI interlingua.

In Xenon, preferences are transformed internally into pseudo-distinctions that have the same form as the corresponding type of distinctions. The distinctions correspond to a particular near-synonym, and also have frequencies — ex-

cept the stylistic distinctions. In this way preferences can be directly compared to distinctions. The pseudo-distinctions corresponding to the previous examples of preferences are:

(- low formality)
(- always high pejorative :agent)
(- always medium implication (C/assessment :MOD (OR ignorant uninformed))).

For each near-synonym w in a cluster, a weight is computed by summing the degree to which the near-synonym satisfies each preference from the set P of input preferences: $Weight(w, P) = \sum_{p \in P} Sat(p, w)$.

The weights are then transformed through an exponential function that normalizes them to be in the interval $[0, 1]$. The exponentials function that we used is: $f(x) = \frac{e^{x^k}}{e-1}$ The main reason this function is exponential is that the differences between final weights of the near-synonyms from a cluster need to be numbers that are comparable with the differences of probabilities from HALogen's language model. The method for choosing the optimal value of k is presented in Section 7.

For a given preference $p \in P$, the degree to which it is satisfied by w is reduced to computing the similarity between each of w's distinctions and a pseudo-distinction $d(p)$ generated from p. The maximum value over i is taken: $Sat(p, w) = max_i\ Sim(d(p), d_i(w))$, where $d_i(w)$ is the i-th distinction of w. We explain the computation of Sim in the next section.

5 Similarity of distinctions

The similarity of two distinctions, or of a distinction and a preference (transformed into a distinction), is computed similarly to (Edmonds 1999):

$$Sim(d_1, d_2) = \begin{cases} Sim_{den}(d_1, d_2) \\ Sim_{att}(d_1, d_2) \\ Sim_{sty}(d_1, d_2) \end{cases} \quad (1)$$

Distinctions are formed out of several components, represented as symbolic values on certain dimensions. In order to compute a numeric score, each symbolic value has to be mapped into a numeric one. The numeric values (see Table 1) are not as important as their relative difference, since all the similarity scores are normalized to the interval $[0, 1]$.

For stylistic distinctions, the degree of similarity is one minus the absolute value of the difference between the style values.

$Sim_{sty}(d_1, d_2) = 1.0 - |Style(d_2) - Style(d_1)|$

For attitudinal distinctions, similarity depends on the frequencies and the attitudes. The similarity of two frequencies is one minus their absolute differences. For the attitudes, their strength is taken into account.

$Sim_{att}(d_1, d_2) = S_{freq}(d_1, d_2) \cdot S_{att}(d_1, d_2)$

Frequency		Indirectness		Attitude		Strength		Style	
never	.0	suggestion	2	pejorative	−2	low	−1	low	0
seldom	.25	implication	5	neutral	0	medium	0	med.	.5
sometimes	.5	denotation	8	favorable	2	high	1	high	1
usually	.75								
always	1.0								

Table 1: *The functions that map symbolic values to numeric values*

$S_{freq}(d_1, d_2) = 1.0 - |Freq(d_2) - Freq(d_1)|$
$S_{att}(d_1, d_2) = 1.0 - |Att(d_2) - Att(d_1)| \ / \ 6$
$Att(d) = Attitude(d) + \text{sgn}(Attitude(d)) \cdot Strength(d)$

The similarity of two denotational distinctions is the product of the similarities of their three components: frequency, indirectness, and conceptual configuration. The first two scores are calculated as for the attitudinal distinctions. The computation of conceptual similarity (S_{con}) will be discussed in the next section.

$Sim_{den}(d_1, d_2) = S_{freq}(d_1, d_2) \cdot S_{lat}(d_1, d_2) \cdot S_{con}(d_1, d_2)$
$S_{lat}(d_1, d_2) = 1.0 - |Lat(d_2) - Lat(d_1)| \ / \ 8$
$Lat(d) = Indirectness(d) + Strength(d)$

Here are some examples of computing the similarity between distinctions:

a. if $d_1 = (lex_1$ low formality) and $d_2 = (lex_2$ medium formality) then $Sim(d_1, d_2) = 1 - |0.5 - 0| = 0.5$

b. if $d_1 = (lex_1$ always medium implication $P_1)$ and $d_2 = (lex_2$ seldom medium suggestion $P_1)$ then $Sim(d_1, d_2) = S_{freq}(d_1, d_2) \cdot S_{lat}(d_1, d_2) \cdot S_{con}(d_1, d_2) = (1 - |0.25 - 1|) \cdot (1 - (2 + 0 + 5 + 0) \ / \ 8) \cdot 1 = 0.03$

6 Similarity of conceptual configurations

Peripheral concepts in Xenon are complex configurations of concepts. The conceptual similarity function S_{con} is in fact the similarity between two interlingual representations t_1 and t_2. Equation 2 computes similarity by simultaneously traversing the two representations.

$$S_{con}(t_1, t_2) = \begin{cases} S(\text{concept}(t_1), \text{concept}(t_2)) \text{ if } N_{1,2} = 0 \\ \alpha S(\text{concept}(t_1), \text{concept}(t_2)) + \beta \frac{1}{N_{1,2}} \sum_{s_1, s_2} S_{con}(s_1, s_2) \end{cases} \quad (2)$$

In equation 2, concept(C), where C is a interlingual representation, is the main concept (or word) in the representation. The first line corresponds to the situation when there are no roles. The second line deals with the roles: shared by both representations, or appearing only in one of them. $N_{1,2}$ is the sum of the number of shared roles and the number of roles unique to each of the representations (at the given level in the interlingua). s_1 and s_2 are the values of any shared role. α and β are weighting factors. If $\alpha = \beta = 0.5$, the whole sub-structure is weighted equally to the main concepts.

The similarity function S deals with the case in which the main concepts are atomic (words or basic concepts) or when they are an OR or AND of complex concepts. If both are disjunctions, $C_1 = (\text{OR } C_{11} \cdots C_{1n})$, and $C_2 = (\text{OR } C_{21} \cdots C_{2m})$, then $S(C_1, C_2) = max_{i,j}\ S_{con}(C_{1i}, C_{2j})$. The components could be atomic or they could be complex concepts; that's why the S_{con} function is called recursively. If one of them is atomic, it can be viewed as a disjunction with one element, so that the previous formula can be used. If both are conjunctions, then the formula computes the maximum of all possible sums of pairwise combinations. If $C_1 = (\text{AND } C_{11}\, C_{12} \cdots C_{1n})$, and $C_2 = (\text{AND } C_{21}\, C_{22} \cdots C_{2m})$, then the longest conjunction is taken. Let's say $n \geq m$ (if not the procedure is similar). All the permutations of the components of C_1 are considered, and paired with components of C_2. If some components of C_1 remain without pair, they are paired with null (and the similarity between an atom and null is zero). Then the similarity of all pairs in a permutation is summed and divided by the number of pairs, and the maximum (from all permutations) is the resulting score: $S(C_1, C_2) = max_{p \in perms}\ \frac{1}{n}\ (\sum_{k=1}^{m} S_{con}(C_{1p_k}, C_{2k}) + \sum_{k=m+1}^{n} S_{con}(C_{1k}, \text{null}))$. Here is a simple example to illustrate this procedure: $S_{con}((\text{AND } a\, b\, c),$ $(\text{AND } b\, a)) = \frac{1}{3}\ max(S_{con}(a, b) + S_{con}(b, a) + S_{con}(c, \text{null}),\ S_{con}(a, a) + S_{con}(b,$ $b) + S_{con}(c, \text{null})) = \frac{1}{3}\ max(0 + 0 + 0, 1 + 1 + 0) = 0.66$

The similarity of two words or two atomic concepts is computed from their positions in the ontology of the system. A simple approach would be this: the similarity is 1 if they are identical, 0 otherwise. But we have to factor in the similarity of two words or concepts that are not identical but closely related in the ontology. We implemented a measure of similarity for all the words, using the Sensus ontology. Two concepts are similar if there is a link of length one or two between them in Sensus. The degree of similarity is discounted by the length of the link. The similarity between a word and a concept is given by the maximum of the similarities between all the concepts (senses) of the word and the given concept. The similarity of two words is given by the maximum similarity between pairs of concepts corresponding to the words. Before looking at the concepts associated with the words, stemming is used to see if the two words share the same stem, in which case the similarity is 1. This enables similarity across parts-of-speech.

Here are some examples of similarity of conceptual configurations:

a. if C_1 = (C1 / departure :MOD physical and :PRE-MOD unusual) and C_2 = (C2 / departure :MOD physical) then $con(C_1, C_2) = 0.5 \cdot 1 + 0.5 \cdot 1/2 \cdot (0.5 \cdot 1) = 0.625$

b. if C_1 = (C1 / person :AGENT_OF (A / drinks :MOD frequently) and then $S_{con}(C_1, C_2) = 0.5 \cdot 1 + 0.5 \cdot 1/1 \cdot (0.5 + 0.5 \cdot 1/2 \cdot 1) = 0.875$ if C_1 = (C1 / occurrence :MOD (OR embarrassing awkward)) and C_2 = (C2 / occurrence :MOD awkward) then $S_{con}(C_1, C_2) = 0.5 \cdot 1 + 0.5 \cdot 1/1 \cdot 1 = 1.0$

c. if C_1 = (C1 / (AND spirit purpose) :MOD hostile) and C_2 = (C2 / purpose :MOD hostile) then $S_{con}(C_1, C_2) = 0.5 \cdot (1 + 0)/2 + 0.5 \cdot 1 = 0.75$

7 Evaluation of Xenon

The main components of Xenon are the near-synonym choice module and HALogen. An evaluation of HALogen was already presented by Langkilde-Geary (2002). Here, we evaluate the near-synonym choice module and its interaction with HALogen.

We conducted two kinds of evaluation experiments. The first type of experiment (Test1 and Test2) feeds Xenon with a suite of inputs: for each test case, an interlingual representation and a set of nuances. The set of nuances correspond to a given near-synonym. A graphic depiction of these two tests is shown in Figure 4. The sentence generated by Xenon is considered correct if the expected near-synonym was chosen. The sentences used in Test1 and Test2 are very simple; therefore, the interlingual representations were easily built by hand. In the interlingual representation, the near-synonym was replaced with the corresponding meta-concept.

The analyzer of lexical nuances for English simply extracts the distinctions associated with a near-synonym in the LKB of NS. Ambiguities are avoided because the near-synonyms in the test sets are members in only one of the clusters used in the evaluation.

In Test1, we used 32 near-synonyms that are members of the 5 clusters; one example of cluster is: *benefit, advantage, favor, gain, profit.* Test1 was used as a development set, to choose the exponent k for the function that translated the scale of the weights. As the value of k increased (staring at 1) the accuracy on the development set increased. The final value chosen for k was 15. In Test2, we used 43 near-synonyms selected from 6 other clusters, namely the English near-synonyms from Figure 5. Test2 was used only for testing, not for development.

The second type of experiment (Test3 and Test4) is based on machine translation. These experiments measure how successful the translation of near-synonyms from French into English and from English into English is. The machine translation experiments were done on French and English sentences that are translations of each other, extracted from the Canadian Hansard (1.3 million pairs of aligned sentences from the official records of the 36th Canadian Parliament) (http://www.

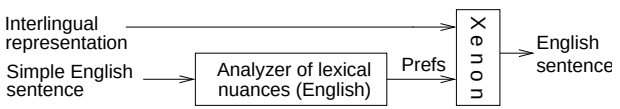

Figure 4: *The architecture of Test1 and Test2*

English (n.): mistake, blooper, blunder, boner, contretemps, error, faux pas, goof, slip, solecism
French: erreur, égarement, illusion, aberration, malentendu, mécompte, bévue, bêtise, blague, gaffe, boulette, brioche, maldonne, sophisme, lapsus, méprise

English (n.): alcoholic, boozer, drunk, drunkard, lush, sot
French: ivrogne, alcoolique, intempérant, dipsomane, poivrot, pochard, sac à vin, soûlard, soûlographe, éthylique, boitout, imbriaque

English (v.): leave, abandon, desert, forsake
French: abandonner, délaisser, déserter, lâcher, laisser tomber, planter là, plaquer, livrer, céder

English (n.): opponent, adversary, antagonist, competitor, enemy, foe, rival
French: ennemi, adversaire, antagoniste, opposant, détracteur

English (adj.): thin, lean, scrawny, skinny, slender, slim, spare, svelte, willowy
French: mince, élancé, svelte, flandrin, grêle, fluet, effilé, fuselé, pincé

English (n.): lie, falsehood, fib, prevarication, rationalization, untruth
French: mensonge, menterie, contrevérité, hâblerie, vanterie, fanfaronnade, craque, bourrage de crâne

Figure 5: *Near-synonyms used in the evaluation of Xenon (Test2,3,4)*

isi.edu/natural-language/download/hansard/). Xenon should generate an English sentence that contains an English near-synonym that best matches the nuances of the initial French near-synonym. If Xenon chooses exactly the English near-synonym used in the parallel text, this means that Xenon's behaviour was correct. This is a conservative evaluation measure, because there are cases in which more than one translation is correct.

The French–English translation experiments take French sentences (that contain near-synonyms of interest) and their equivalent English translations. We can assume that the interlingual representation is the same for the two sentences. Therefore, we can use the interlingual representation for the English sentence to approximate the interlingual representation for the French sentence. As depicted in Figure 6, the interlingual representation is produced using an existing tool that was previously used by Langkilde-Geary (2002) in HALogen's evaluation experiments. In order to use this input construction tool, we parsed the English

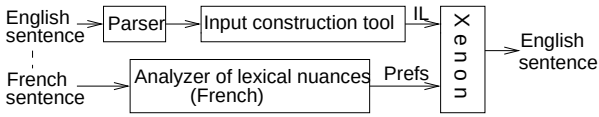

Figure 6: *The architecture of Test3 and Test4, the French-to-English part*

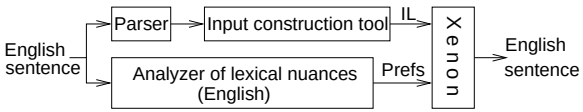

Figure 7: *The architecture of Test3 and Test4, the English-to-English part*

sentences with Charniak's parser (Charniak 2000), and we annotated the subjects in the parse trees. In the interlingual representation produced by the input construction tool we replaced the near-synonym with the corresponding meta-concept. For Test4, the baseline is much higher because of the way the test data was collected. The analyzer of French nuances from Figure 6 needs to extract nuances from a LKB of French synonyms. We created by hand a LKB for six clusters of French near-synonyms, from two paper dictionaries of French synonyms, (Bénac 1956) and (Bailly 1973).

A similar experiment, translating not from French into English but from English into English via the interlingual representation, is useful for evaluation purposes. An English sentence is processed to obtain its semantic representation (and the near-synonym is replaced with a meta-concept), and the lexical nuances of the near-synonym are fed as preferences to Xenon. Ideally, the same near-synonym as in the original sentence would be chosen by Xenon. The percentage of times this happens is an evaluation measure. The architecture of these experiments is presented in Figure 7.

Test4 is similar to Test3, but instead of having one sentence for a near-synonym, it contains all the sentences in a given fragment of Hansard (4 Megabytes of House debates) in which words of interest occurred. There fore, Test4 has the advantage of containing naturally occurring text, not artificially selected sentences. It has the disadvantage of lacking those near-synonyms in the test set that are less frequent. The English and French near-synonyms used in Test3 and Test4 are the ones presented in Figure 5.

In order to measure the baseline (the performance that can be achieved without using the LKB of NS), we ran Xenon on all the test cases, but this time with no input preferences. Some of the choices could be correct solely because the expected near-synonym happens to be the default one.

The results of the evaluation experiments are presented in Table 2. The second column shows the number of test cases. The column named "Correct" shows the number of answers considered correct. The column named "Ties" shows the number of ties, that is, cases when the expected near-synonym had weight 1.0, but there were other near-synonyms that also had weight 1.0, because they happen to have the same nuances in the LKB of NS. In such cases, Xenon cannot be expected to make the correct choice. More precisely, the other choices are equally correct, at least as far as Xenon's LKB is concerned. Therefore, the accuracies computed without considering these cases (the seventh column) are

Experiment	No. of cases	Correct by default	Correct	Ties	Baseline %	Accuracy (no ties) %	Acc. (total) %
Test1 Simple, dev.	32	5	27	4	15.6	84.3	96.8
Test2 Simple, test	43	6	35	5	13.9	81.3	93.0
Test3 Fr-En, test	14	5	7	2	35.7	50.0	64.2
Test3 En-En, test	14	5	14	0	35.7	100	100
Test4 Fr-En, test	50	37	39	0	76.0	78.0	78.0
Test4 En-En, test	50	37	49	0	76.0	98.0	98.0

Table 2: *Xenon evaluation experiments and their results*

underestimates of the real accuracy of Xenon. The eighth column presents accuracies while taking the ties into account, defined as the number of correct answers divided by the difference between the number of cases and the number of ties.

There are two reasons to expect Xenon's accuracy to be less then 100%. The first is that there are cases in which two or more near-synonyms get an equal, maximal score because they do not have any nuances that differentiate them (either they are perfectly interchangeable, or the LKB of NS does not contain enough knowledge) and the one chosen is not the expected one. The second reason is that sometimes Xenon does not choose the expected near-synonym even if it is the only one with maximal weight. This may happen because HALogen makes the final choice by combining the weight received from the near-synonym choice module with the probabilities from its language model. Frequent words may have high probabilities in the language model. If the expected near-synonym is very rare, its probability is very low. When combining the weights with the probabilities, a frequent near-synonym may win even if it has a lower weight assigned by the near-synonym choice module. In such cases, the default near-synonym (the most frequent of the cluster) wins. Sometimes such a behaviour is justified, because there may be other constraints that influence HALogen's choice.

Table 2 also includes the results for the baseline experiments. For Test1 and Test2 the baseline is much lower than Xenon's performance. For example, for Test1, Xenon achieves 96.8% accuracy, while the baseline is 15.6%. An exception is the baseline for Test4, which is high (76%), due to the way the test data was collected: it contains sentences with frequent near-synonyms, which happen to be the ones Xenon chooses by default in the absence of input preferences. For Test3 and Test4 the baseline is the same for the French and English experiments because there no nuances were used.

Another way to measure the performance of Xenon is to measure how many times it makes appropriate choices that cannot be made by HALogen, that is, cases that make good use of the nuances from the LKB of NS. This excludes the test cases with default near-synonyms, for which Xenon makes the right choice due to the language model. It also excludes the cases of ties when Xenon cannot

make a choice. For the experiments with only English near-synonyms, Xenon is performing very well, managing to make correct choices that cannot be made by default. Accuracies vary from 92.3% to 100%. For the experiments involving both French and English experiments, Xenon makes only a few correct choices that cannot be made by default. This is due to the fact that there is some overlap in nuances between French and English synonyms, but most of the overlap happens for the near-synonyms that are defaults.

8 Conclusion

Xenon can successfully choose the near-synonym that best matches a set of input preferences. In recent work, we extended the near-synonym choice to take into account the collocational behaviour of the near-synonyms.

Acknowledgments. We thank Irene Langkilde-Geary for useful discussions and for making the input construction tool available. We thank Phil Edmonds for making available the source code of I-Saurus. Our work is financially supported by the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

Bailly, René, ed. 1973. *Dictionnaire des Synonymes de la Langue Française*. Paris: Larousse.

Bénac, Henri, ed. 1956. *Dictionnaire des Synonymes*. Paris: Librarie Hachette.

Charniak, Eugene. 2000. "A Maximum-Entropy-Inspired Parser". *The 1st Conference of the North American Chapter of the Association for Computational Linguistics and the 6th Conference on Applied Natural Language Processing* (*NAACL-ANLP 2000*), 132-139. Seattle, Washington, U.S.A.

Edmonds, Philip & Graeme Hirst. 2002. "Near-Synonymy and Lexical Choice". *Computational Linguistics* 28:2.105-145.

Edmonds, Philip. 1999. *Semantic Representations of Near-Synonyms for Automatic Lexical Choice*. Ph.D. dissertation, University of Toronto.

Hayakawa, Samuel I., 1994. *Choose the Right Word.* Second Edition, ed. by Eugene Ehrlich. New York: Harper Collins.

Inkpen, Diana Zaiu & Graeme Hirst. 2001. "Building a Lexical Knowledge-Base of Near-Synonym Differences". *Workshop on WordNet and Other Lexical Resources, Second Meeting of the North American Chapter of the Association for Computational Linguistics* (*NAACL 2001*), 47-52. Pittsburgh, U.S.A.

Inkpen, Diana. 2004. *Building a Lexical Knowledge-Base of Near-Synonym Differences*. Ph.D. dissertation, University of Toronto.

Knight, Kevin & Steve Luk. 1994. "Building a Large Knowledge Base for Machine Translation". *The 12th National Conference on Artificial Intelligence* (*AAAI-94*), 773-778. Seattle, Washington, U.S.A.

Langkilde, Irene & Kevin Knight. 1998. "Generation that Exploits Corpus-Based Statistical Knowledge". *The 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics* (*ACL-COLING'98*), 704-712. Montréal, Québec, Canada.

Langkilde-Geary, Irene. 2002. "An Empirical Verification of Coverage and Correctness for a General-Purpose Sentence Generator". *The 12th International Natural Language Generation Workshop*, 17-24. New York, U.S.A.

Fast and Accurate Part–of–Speech Tagging: The SVM Approach Revisited

JESÚS GIMÉNEZ & LLUÍS MÀRQUEZ
Technical University of Catalonia

Abstract

A very simple and effective part-of-speech tagger based on Support Vector Machines (SVMT) is presented. Simplicity and efficiency are achieved by working with linear separators in the primal formulation of SVM, using a greedy tagging scheme. By means of a rigorous experimental evaluation, we conclude that the proposed SVM-based tagger is robust and flexible for feature modelling, trains efficiently with almost no parameters to tune, and is able to tag thousands of words per second, which makes it suitable for real applications. Regarding accuracy, the SVM tagger significantly outperforms the TnT tagger, and achieves an accuracy of 97.2% on the WSJ corpus, which is comparable to the best taggers reported up to date.

1 Introduction

Automatic part–of–speech (POS) tagging is the task of determining the morphosyntactic category of each word in a given sentence. It is a very well-known problem that has been addressed by many researchers for the last decades. It is a *fundamental* problem in the sense that almost all NLPapplications need some kind of POS tagging previous to construct more complex analysis and it is permanently *on–fashion* since current applications demand an efficient treatment of more and more quantities of text.

In the recent literature, we can find several approaches to POS tagging based on statistical and machine learning techniques, including among many others: Hidden Markov Models (Brants 2000), Maximum Entropy (Ratnaparkhi 1996), Transformation–based learning (Brill 1995), Memory–based learning (Daelemans et al. 1996), Decision Trees (Màrquez et al. 1999), AdaBoost (Abney et al. 1999), and Support Vector Machines (Nakagawa et al. 2001). Most of the previous taggers have been evaluated on the English WSJ corpus, using the Penn Treebank set of POS categories and a lexicon constructed directly from the annotated corpus. Although the evaluations were performed with slight variations, there was a wide consensous in the late 90's that the state–of–the–art accuracy for English POS tagging was between 96.4% and 96.7%. The most widespread taggers in the NLP community have been the HMM–based TnT tagger (Brants 2000), the Transformation–based learning (TBL) tagger (Brill 1995), and several variants of the Maximum Entropy (ME) approach (Ratnaparkhi 1996). TnT is an example of a really practical tagger for NLP applications. It is available to

anybody, simple and easy to use, considerably accurate, and extremely efficient. In the case of TBLand ME approaches, the great success has been due to the flexibility they offer in modelling contextual information.

Far from being considered a closed problem, researchers have tried to improve results during last years. Either by introducing more complex HMM models (Lee et al. 2000), or by enriching the feature set in a ME tagger (Toutanova & Manning 2000), or by using more effective learning techniques: SVM (Nakagawa et al. 2001), a Voted–Perceptron–based training of a ME model (Collins 2002), and Cyclic Dependency Networks (Toutanova et al. 2003). The state–of–the–art accuracy was raised up to 96.9%–97.2% on the same WSJ corpus. In a complementary direction, other researchers suggested the combination of several pre-existing taggers under several alternative voting schemes (Brill & Wu 1998, van Halteren et al. 1998). The accuracy of these taggers was also around 97.2%.

We propose to go back to the TnT *philosophy* (i.e., simplicity and efficiency with state–of–the–art accuracy) but within the SVM learning framework. We claim that the SVM–based tagger introduced in this work fulfills the requirements for being a practical tagger and offers a very good balance of the following properties. (1) *Simplicity*: the tagger is easy to use and has few parameters to tune; (2) *Flexibility* and *robustness*: rich context features can be efficiently handled without overfitting problems, allowing lexicalization; (3) *High accuracy*: the SVM–based tagger performs significantly better than TnT and achieves an accuracy competitive to the best current taggers; (4) *Efficiency*: training on the WSJ is performed in around one CPU hour and the tagging speed allows a massive processing.

It is worth noting that the SVM paradigm has been already applied to tagging in a previous work (Nakagawa et al. 2001), with the focus on the guessing of unknown word categories. The final tagger constructed in that paper gave a clear evidence that the SVM approach is specially appropriate for the second and third of the previous points, the main drawback being a low efficiency (a running speed of around 20 words per second was reported). We overcome this limitation by working with linear kernels in the primal setting of the SVM framework taking advantage of the extremely sparsity of example vectors. The resulting tagger is as accurate as that of (Nakagawa et al. 2001) but 60 times faster in a Perl prototype.

2 Support Vector Machines

SVM is a machine learning algorithm for binary classification, which has been successfully applied to a number of practical problems, including NLP. See (Cristianini & Shawe–Taylor 2000) for a good introduction.

Let $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ be the set of N training examples, where each instance $\mathbf{x}_i$ is a vector in $\mathbb{R}^M$ and $y_i \in \{-1, +1\}$ is the class label. In their basic form, a SVM learns a linear hyperplane that separates the set of positive examples

from the set of negative examples with *maximal margin* (the margin is defined as the distance between the hyperplane and the nearest of the positive and negative examples). This learning bias is a heuristic intended to minimize the bound on generalization error.

The linear separator is defined by two elements: a weight vector $\mathbf{w}$ and a bias term b. Given a new example $\mathbf{x}$, the classification rule of a SVM is $sgn(f(\mathbf{x}, \mathbf{w}, b))$, where $f(\mathbf{x}, \mathbf{w}, b) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$. In the linearly separable case, learning the maximal margin hyperplane $(\mathbf{w}, b)$ can be stated as a convex quadratic optimization problem with a unique solution: *minimize* $||\mathbf{w}||$, *subject to the constraints* (one for each training example): $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1$.

The SVM model has an equivalent dual formulation, characterized by a weight vector $\boldsymbol{\alpha}$ and a bias b. In this case, $\boldsymbol{\alpha}$ contains one weight for each training vector, indicating the importance of this vector in the solution. Vectors with non null weights are called *support vectors*. The dual classification rule is: $f(\mathbf{x}, \boldsymbol{\alpha}, b) = \sum_{i=1}^{N} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b$, and the $\boldsymbol{\alpha}$ vector can be calculated also as a quadratic optimization problem. Given the optimal $\boldsymbol{\alpha}^*$ vector of the dual quadratic optimization problem, the weight vector $\mathbf{w}^*$ that realizes the maximal margin hyperplane is calculated as $\mathbf{w}^* = \sum_{i=1}^{N} y_i \alpha_i^* \mathbf{x_i}$.
The b^* has also a simple expression in terms of $\mathbf{w}^*$ and the training examples.

The advantage of the dual formulation is that it permits an efficient learning of non–linear SVM separators, by introducing *kernel functions*. A kernel function calculates a dot product between two vectors that have been (non linearly) mapped into a high dimensional feature space. In spite of the high dimensionality of the real feature space, the training is still feasible, since there is no need to perform this mapping explicitly.

In the presence of outliers and wrongly classified examples it may be useful to allow some training errors in order to avoid overfitting. This is achieved by a variant of the optimization problem, referred to as *soft margin*, in which the contribution of errors to the objective function of margin maximization can be balanced through the use of the C parameter.

3 Problem setting

Tagging a word in context is a multi–class classification problem. We have applied a simple *one–per–class* binarization, i.e., a SVM is trained for every part–of–speech in order to distinguish between examples of this class and all the rest. When tagging a word the most confident tag according to the predictions of all SVMs is selected.

However, not all training examples have been considered for all classes. Instead, a dictionary is extracted from the training corpus with all possible tags for each word, and when considering the occurrence of a training word w tagged as t_i, this example is used as a positive example for class t_i and a negative exam-

word unigrams, bigrams and trigrams	$w_{-3}, w_{-2}, w_{-1}, w_0, w_{+1}, w_{+2}, w_{+3}$, $(w_{-2}, w_{-1}), (w_{-1}, w_{+1}), (w_{-1}, w_0), (w_0, w_{+1}), (w_{+1}, w_{+2})$, $(w_{-3}, w_{-2}, w_{-1}), (w_{-2}, w_{-1}, w_0), (w_{-2}, w_{-1}, w_{+1})$, $(w_{-1}, w_0, w_{+1}), (w_{-1}, w_{+1}, w_{+2}), (w_0, w_{+1}, w_{+2})$
POS unigrams, bigrams and trigrams	$p_{-3}, p_{-2}, p_{-1}, (p_{-2}, p_{-1}), (p_{-1}, a_{+1}), (a_{+1}, a_{+2})$ $(p_{-3}, p_{-2}, p_{-1}), (p_{-2}, p_{-1}, a_{+1}), (p_{-1}, a_{+1}, a_{+2})$
ambiguity classes, may_be's	$a_0, a_1, a_2, a_3, m_0, m_1, m_2, m_3$

Table 1: *Feature patterns used to codify examples*

ple for all other t_j classes appearing as possible tags for w in the dictionary. In this way, we avoid the generation of excessive (and irrelevant) negative examples, and we make the training step faster. In the following sections we will see how a 635,000 word corpus generates training sets of about 20,000 examples on average, instead of 635,000.

Each example has been codified on the basis of the *local context* of the word to be disambiguated. We have considered a centered window of seven tokens, in which some basic and n–gram patterns are evaluated to form binary features such as: "previous_word_is_`the`", "two_preceeding_tags_are_`DT_NN`", etc. Table 1 contains the list of all patterns considered.

As it can be seen, the tagger is lexicalized and all word forms appearing in window are taken into account. Since a very simple left–to–right tagging scheme will be used, word tags are not known at running time. Following the approach of (Daelemans et al. 1996) we use the more general *ambiguity–class* tag for the right context words, which is a label composed by the concatenation of all possible tags for the word (e.g., `IN-RB`, `JJ-NN`, etc.). Each of the individual tags of an ambiguity class is also taken as a binary feature of the form "following_word_**may_be**_a_`VBZ`". We avoid the *two passes* solution proposed in (Nakagawa et al. 2001), in which a first tagging is performed in order to have right contexts disambiguated for the second pass. They suggested to replace the use of explicit n–gram features by applying polynomial kernels. However, since we are interested in working with a linear kernel, we have included them in the feature set.

4 Experiments

The Wall Street Journal data from the Penn Treebank III has been used as the benchmark corpus. We have randomly divided this 1,17 million word corpus into three subsets: training (60%), validation (20%) words, and test (20%). All the tagging experiments reported are evaluated on the complete test set. The validation set has been used to optimize parameters. The Penn Treebank tagset contains 48 tags. However, after compiling training examples in the way explained in section 3, only 34 of them receive positive and negative examples. Thus, only 34 SVM classifiers have to be trained in the binarized setting. The 14

model	acc.	#sv	l_time
$d = 1$, fs_1	93.50%	3,472.47	44':48"
$d = 2$, fs_1	93.91%	4,385.56	4h:10':19"
$d = 3$, fs_1	93.47%	6,040.59	5h:35':30"
$d = 4$, fs_1	92.78%	8,196.74	7h:23':07"
$d = 1$, fs_2	93.84%	3,532.44	1h:21':14"
$d = 2$, fs_2	93.67%	4,935.74	5h:38':48"

Table 2: *Accuracy results on ambiguous words of alternative SVMT models*

unambiguous tags correspond to punctuation marks, symbols, and the categories `TO` and `WP$`.

4.1 *Linear vs. polynomial kernels*

So as to evaluate the effect of the kernel in the training process and in the generalization accuracy, we have trained several SVM classification models with the whole 635k word training set by varying the degree (d) of the polynomial kernel and the feature set (fs). The software package used in all the experiments reported was SVM^{light}.[1] A very simple frequency threshold was used to filter out unfrequent features. In particular, we have discarded features that occur less than n times, where $n \geq 2$ is the minimum number so that the total amount of features is not greater than 100,000. When training, the setting of the C parameter has been left to its default value. In the following subsections we will see how the optimization of this parameter leads to a small improvement in the final tagger, being the SVM algorithm quite robust with respect to parameterization.

Results obtained when classifying the ambiguous words in the test set are presented in Table 2. '#sv' stands for the average number of support vectors per tag and 'l_time' is the CPU training time. This test has been performed in a *batch* mode, that is, examples of ambiguous words are taken separately, and the features involving the POS tags of the left contexts are calculated using the *correct* POS tag assignment of the corpus.

When using the set of atomic features (fs_1), the best results are obtained with a degree 2 polynomial kernel (93.91%). Greater degrees produce overfitting, since the number of support vectors highly increases and the accuracy decreases.

When using the n-gram extended set of features (fs_2), the linear kernel improves from 93.50% to 93.84% and becomes really competitive to the degree 2 polynomial kernel on fs_1 (93.90%). Actually, it is clearly preferable due to the higher sparsity of the solution and the shorter learning time. Interestingly, the advantage of the extended set of features is only noticeable in the case of the linear kernel, since accuracy decreases when used with the degree 2 polynomial kernels.

[1] The SVM^{light} software is freely available at `http://svmlight.joachims.org`.

words	amb.%	all%	#exs	#feat.	#sv	#$\mathbf{x}_i$	#$\mathbf{w}$	l_time	t_time
50k	90.53	96.27	1,572.1	38,613	500.4	38.00	3,339.4	05':12"	3':21"
100k	91.90	96.81	3,141.8	72,030	879.8	38.03	5,552.2	10':52"	3':26"
200k	92.71	97.13	6,237.8	50,699	1,424.4	38.02	5,911.0	21':26"	3':26"
300k	93.15	97.29	9,371.4	72,988	1,959.0	38.01	7,833.8	34':16"	3':33"
400k	93.45	97.40	12,501.9	93,660	2,490.1	38.02	9,669.2	47':07"	3':36"
500k	93.67	97.48	15,672.4	89,179	2,935.4	38.04	10,102.8	1h:00':12"	3':38"
600k	93.74	97.52	17,875.2	86,273	3,218.9	38.04	10,297.5	1h:16':15"	3':36"
635k	93.84	97.56	19,951.6	90,517	3,556.1	38.04	11,041.6	1h:21':14"	3':37"

Table 3: *Accuracy results of SVMT under the closed vocabulary assumption*

As a conclusion, we can state that a linear kernel, with an n–gram based set of basic features, suffices to obtain highly accurate SVM models relatively fast to train. In the next section we will see how the linear solution has the additional advantage of allowing to work in the primal setting with a very sparse vector of weights. This fact is crucial to obtain a fast tagger.

4.2 *Evaluating the SVM tagger*

Hereafter we will focus only on the linear SVM model. In this experiment, the POS tagger is tested in a more realistic situation, that is, performing a left–to–right tagging of the sequence of words, with an *on-line* calculation of features making use of the already assigned left-context POS tags. Following the simplicity and efficiency principles, a greedy left–to–right tagging scheme is applied, in which no optimization of the tag sequence is performed at the sentence level. We have implemented this first POS tagger prototype in Perl-5.0, which will be referred to as SVMT. The SVM dual representation based on a set of support vectors output by SVM^{light} is converted into the primal form $(\mathbf{w}, b)$ using the equation explained in section 2.

The tagger has been tested under the *closed vocabulary assumption*, in which no unknown words are considered. This is simulated by directly including in the dictionary the words of the test set that do not occur in the training set. The results obtained by increasing sizes of the training set are presented in Table 3. 'amb.' and 'all' columns contain accuracy achieved on ambiguous words and overall, respectively. '#exs' and '#sv' stand for the average number of examples and support vectors per POS tag, '#feat.' for the total number of binary features (after filtering), '#$\mathbf{x}_i$' for the average number of active features in the training examples, and '#$\mathbf{w}$' for the average number of dimensions of the weight vectors. 'l_time' and 't_time' refer to learning and tagging time. Learning and tagging times are also graphically presented in Figure 1. [2]

As it could be expected, the accuracy of the tagger grows with the size of the training set, presenting a logarithmic behaviour. Regarding efficiency, it can be

[2] Experiments were performed using a 2Ghz Pentium-IV processor with 1Gb of RAM.

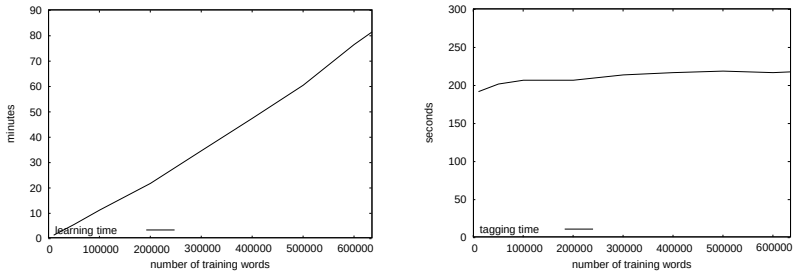

Figure 1: *Learning and tagging time plots of SVMT by increasing sizes of the training set*

observed (Figure 1, left plot) that training time is almost linear with respect to the number of examples of the training set. Besides, the compression of the SVM model with respect to the training set also increases with the training set size, and goes from 68.17% (from 1572.1 examples to 500.4 support vectors) to 82.18% (from 19,951.6 examples to 3,556.1 support vectors). However, this compression level would not permit an efficient tagger in the dual form, since thousands of dot products would be needed to classify each word. More interestingly, the model in primal form is quite compact since the weight vectors, resulting from compacting all support vectors, contains no more than 11,000 features among the 90,000 possible features. Provided that the test examples are very sparse, the classification rule is very efficient, since a single dot product with a sparse vector is needed to classify each word. Basing this dot product on the non null dimensions of the example to classify, the tagging time can be mantained almost invariant (see Figure 1, right plot).

4.3 *Including unknown words*

The tagger results presented in the previous section are still not realistic, since we can not assume a closed vocabulary. In order to deal with this problem we have developed a SVM-based model to recognize unknown words. Unknown words are treated as ambiguous words with all possible POS tags corresponding to open-class words (18 in the Penn Treebank tagset), but specialized SVMs are learned with particular features to disambiguate among the POS tags of unknown words. The approach is similar to that of (Nakagawa et al. 2001) and the features used, which are presented in table 4, are inspired in (Brill 1995, Màrquez et al. 1999, Nakagawa et al. 2001) and consist of ortographic information such as prefixes, suffixes, capitalization, etc.

Results obtained are presented in Table 5. ‘known’ and ‘unk.’ refer to the subsets of known and unknown words respectively, ‘amb’ to the subset of ambiguous known words, and ‘all’ to the overall accuracy. The label SVMT+ corre-

All features for known words	(see table 1)
prefixes	s_1, $s_1 s_2$, $s_1 s_2 s_3$, $s_1 s_2 s_3 s_4$
suffixes	s_n, $s_{n\text{-}1} s_n$, $s_{n\text{-}2} s_{n\text{-}1} s_n$, $s_{n\text{-}3} s_{n\text{-}2} s_{n\text{-}1} s_n$
begins with Upper Case	yes/no
all Upper Case	yes/no
all Lower Case	yes/no
contains a Capital Letter not at the beginning	yes/no
contains more than one Capital Letter not at the beginning	yes/no
contains a period	yes/no
contains a number	yes/no
contains a hyphen	yes/no
word length	integer

Table 4: *Feature templates for unknown words*

system	amb.	known	unk.	all
TnT	92.2%	96.9%	84.6%	96.5%
SVMT	93.6%	97.2%	83.5%	96.9%
SVMT+	94.1%	97.3%	83.6%	97.0%

Table 5: *Accuracy results of SVMT compared to TnT*

sponds to the case in which the C parameter has been tuned.[3]

Similar to the previous section the results obtained by the SVMT clearly outperform the results of TnT tagger and they are comparable to the accuracy of the best current taggers (which range from 96.9% to 97.1%). Again, the tuning of the C parameter provides a small increase of performance, but at a cost of increasing the training time to almost 20 CPU hours.

The Perl implementation of SVMT+ model achieves a tagging speed of 1,335 words per second. Given the type of operations computed by the tagging algorithm we fairly believe that a re-implementation in C++ could speed up the tagger, making the efficiency valid for massive text processing. Of course, the TnT tagger is still much more efficient, achieving a tagging speed of more than 50,000 words per second on the same conditions.

5 Discussion

The work presented is closely related to (Nakagawa et al. 2001). In that paper, a SVM–based tagger is presented and compared to TnT, obtaining a best accuracy of 97.1% (when training from a 1 million word corpus). Apart from the training set size, the main differences between both approaches are the following. Nakagawa's work is focused on tagging unknown words. A certainly *ad–hoc* proce-

[3] The C parameter value is 0.109 for known words and 0.096 for unknown words.

dure is performed to tag the word sequence in two passes. In the first pass, the tagger disambiguates the whole sentence and in the second pass, the previously assigned tags are assumed correct in order to extract the right–context tag features. This tagging overhead is also projected into training, since two versions, with and without right–context tag features, must be trained. Additionally, it is argued that one advantage of using SVM is that it is not necessary to codify n–gram features, since polynomial kernels themselves succeed at doing this task. Thus, they base their best tagger in a dual solution with degree 2 polynomial kernels, instead of a linear separator in the primal setting.

Since they are using kernels and SVM classification in the dual setting, the tagger simply can not be fast at running time. In particular, a tagging speed of 19.80 words per second (4 hours needed to tag a 285k word test set) is reported. Training is also much more slower than ours (the time required for training from a 100k word set is about 16.5 hours), probably due to the way in which they select training examples for each POS.

6 Conclusions and future work

In this work we have presented a SVM–based POS tagger suitable for real applications, since it provides a very good balance of several good properties for NLP tools:simplicity, flexibility, high performance, and efficiency. The next step we plan to do is to re-implement the tagger in C++ to significantly increase efficiency and to provide a software package for public use.[4]

Regarding the study of the SVM–approach some issues deserve further investigation. First, the learning model for unknown words experimented so far is preliminary and we think that can be clearly improved. Second, we have applied the simplest greedy tagging scheme. Since SVM predictions can be converted into confidence values, a natural extension would be to consider a sentence–level tagging model in which the confidence value of the whole sentence assignment is maximized. Finally, unsupervised learning techniques are being studied and implemented so as to allow SVMT to be suitable for languages for which no labeled data is available.

Acknowledgements. This research has been partially funded by the Spanish Ministry of Science and Technology (MCyT's projects: HERMES TIC2000-0335-C03-02, ALIADO TIC2002-04447-C02) and by the European Comission (LC-STAR IST-2001-32216), and by the Catalan Research Department (CIRIT's consolidated research group 2001SGR-00254).

[4] SVMT is available for public demostration at: `www.lsi.upc.es/~nlp/SVMTool`.

REFERENCES

Abney, S., R. E. Schapire & Y. Singer. 1999. "Boosting Applied to Tagging and PP–attachment". *Proceedings of the 4th Conference on Empirical Methods in Natural Language Processing* (*EMNLP'99*), 38-45. Maryland, U.S.A.

Brants, T. 2000. "TnT - A Statistical Part-of-Speech Tagger". *6th Conference on Applied Natural Language Processing*, 224-231. Seattle, Washington.

Brill, E. 1995. "Transformation–based Error–driven Learning and Natural Language Processing: A Case Study in Part–of–speech Tagging". *Computational Linguistics* 21:4.543-565.

Brill, E. & J. Wu. 1998. "Classifier Combination for Improved Lexical Disambiguation". *17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics* (*ACL-COLING'98*), 191-195. Montréal, Québec, Canada.

Collins, M. 2002. "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms". *7th Conference on Empirical Methods in Natural Language Processing*, 1-8. Philadelphia, Penn.

Cristianini, N. & J. Shawe-Taylor. 2000. *An Introduction to Support Vector Machines*. Cambridge: Cambridge University Press.

Daelemans, W., J. Zavrel, P. Berck & S. Gillis. 1996. "MBT: A Memory-Based Part-of-speech Tagger Generator". *4th Workshop on Very Large Corpora*, 14-27. Copenhagen, Denmark.

Halteren, H. van, J. Zavrel & W. Daelemans. 1998. "Improving Data Driven Wordclass Tagging by System Combination". *17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, 491-497. Montréal, Québec, Canada.

Lee, S., J. Tsujii & H. Rim. 2000. "Part-of-Speech Tagging Based on Hidden Markov Model Assuming Joint Independence". *38th Annual Meeting of the Association for Computational Linguistics* (*ACL'00*), 263-269. Hong Kong.

Màrquez, L., H. Rodríguez, J. Carmona & J. Montolio. 1999. "Improving POS Tagging Using Machine–Learning Techniques". *4th Conference on Empirical Methods in Natural Language Processing*, 53-62. Maryland, U.S.A.

Nakagawa, T., T. Kudoh & Y. Matsumoto. 2001. "Unknown Word Guessing and Part-of-Speech Tagging Using Support Vector Machines". *6th Natural Language Processing Pacific Rim Symposium*, 325-331. Tokyo, Japan.

Ratnaparkhi, A. 1996. "A Maximum Entropy Part-of-speech Tagger". *1st Conference on Empirical Methods in Natural Language Processing*, 133-142. Maryland, U.S.A.

Toutanova, K., & C. D. Manning. 2000. "Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger". *5th Conference on Empirical Methods in Natural Language Processing* (*EMNLP-2000*), 63-71. Hong Kong.

Toutanova, K., D. Klein & C. D. Manning. 2003. "Feature–Rich Part-of-Speech Tagging with a Cyclic Dependency Network". *Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics Conference Series*, 252-259. Edmonton, Canada.

Part-of-Speech Tagging with Minimal Lexicalization

VIRGINIA SAVOVA* & LEONID PESHKIN**

*Johns Hopkins University
**Massachusetts Institute of Technology

Abstract

We use a Dynamic Bayesian Network (DBN) to represent compactly a variety of sublexical and contextual features relevant to Part-of-Speech (POS) tagging. The outcome is a flexible tagger (LegoTag) with state-of-the-art performance (3.6% error on a benchmark corpus). We explore the effect of eliminating redundancy and radically reducing the size of feature vocabularies. We find that a small but linguistically motivated set of suffixes results in improved cross-corpora generalization. We also show that a minimal lexicon limited to function words is sufficient to ensure reasonable performance.

1 Part-of-Speech Tagging

Many NLP applications are faced with the dilemma whether to use statistically extracted or expert-selected features. There are good arguments in support of either view. Statistical feature selection does not require extensive use of human domain knowledge, while feature sets chosen by experts are more economical and generalize better to novel data.

Most currently available POS perform with a high degree of accuracy. However, it appears that the success in performance can be overwhelmingly attributed to an across-the-board lexicalization of the task. Indeed, Charniak, Hendrickson, Jacobson & Perkowitz (1993) note that a simple strategy of picking the most likely tag for each word in a text leads to 90% accuracy. If so, it is not surprising that taggers using vocabulary lists, with number of entries ranging from 20k to 45k, perform well. Even though a unigram model achieves an overall accuracy of 90%, it relies heavily on lexical information and is next to useless on nonstandard texts that contain lots of domain-specific terminology.

The lexicalization of the POS tagging task comes at a price. Since word lists are assembled from the training corpus, they hamper generalization across corpora. In our experience, taggers trained on the Wall Street Journal (WSJ) perform poorly on novel text such as email or newsgroup messages (a.k.a. Netlingo). At the same time, alternative training data are scarce and expensive to create. This paper explores an alternative to lexicalization. Using linguistic knowledge, we construct a minimalist tagger with a small but efficient feature set, which maintains a reasonable performance across corpora.

A look at the previous work on this task reveals that the unigram model is at the core of even the most sophisticated taggers. The best-known rulebased tagger (Brill 1994) works in two stages: it assigns the most likely tag to each word in the text; then, it applies transformation rules of the form "Replace tag X by tag Y in triggering environment Z". The triggering environments span up to three sequential tokens in each direction and refer to words, tags or properties of words within the region. The Brill tagger achieves less than 3.5% error on the Wall Street Journal (WSJ) corpus. However, its performance depends on a comprehensive vocabulary (70k words).

Statistical tagging is a classic application of Markov Models (MMs). Brants (2000) argues that second-order MMs can also achieve state-of-the-art accuracy, provided they are supplemented by smoothing techniques and mechanisms to handle unknown words. TnT handles unknown words by estimating the tag probability given the suffix of the unknown word and its capitalization. The reported 3.3% error for Trigrams 'n Tags (TnT) tagger on the WSJ (trained on 10^6 words and tested on 10^4) appears to be a result of overfitting. Indeed, this is the maximum performance obtained by training TnT until only 2.9% of words are unknown in the test corpus. A simple examination of WSJ shows that such percentage of unknown words in the testing section (10% of WSJ corpus) requires simply building a unreasonably large lexicon of nearly all (about 44k) words seen in the training section (90% of WSJ), thus ignoring the danger of overfitting. Hidden MMs (HMMs) are trained on a dictionary with information about the possible POS of words (Jelinek 1985; Kupiec 1992). This means HMM taggers also rely heavily on lexical information.

Obviously, POS tags depend on a variety of sublexical features, as well as on the likelihood of tag/tag and tag/word sequences. In general, all existing taggers have incorporated such information to some degree. The Conditional Random Fields (CRF) model (Lafferty, McCallum & Pereira 2002) outperforms the HMM tagger on unknown words by extensively relying on orthographic and morphological features. It checks whether the first character of a word is capitalized or numeric; it also registers the presence of a hyphen and morphologically relevant suffixes (*~ed, ~ly, ~s, ~ion, ~tion, ~ity, ~ies*). The authors note that CRF-based taggers are potentially flexible because they can be combined with feature induction algorithms. However, training is complex (AdaBoost + Forward-backward) and slow (10^3 iterations with optimized initial parameter vector; fails to converge with unbiased initial conditions). It is unclear what the relative contribution of features is in this model.

The Maximum Entropy tagger (MaxEnt, see Ratnaparkhi 1996) accounts for the joint distribution of POS tags and features of a sentence with an exponential model. Its features are along the lines of the CRF model:

(1) **Capitalization**: Does the token contain a capital letter?;
(2) **Hyphenation**: Does the token contain a hyphen?;
(3) **Numeric**: Does the token contain a number?;
(4) **Prefix**: Frequent prefixes, up to 4 letters long;
(5) **Suffix**: Frequent suffixes, up to 4 letters long;

In addition, Ratnaparkhi uses lexical information on frequent words in the context of five words. The sizes of the current word, prefix, and suffix lists were 6458, 3602 and 2925, respectively. These are supplemented by special Previous Word vocabularies. Features frequently observed in a training corpus are selected from a candidate feature pool. The parameters of the model are estimated using the computationally intensive procedure of Generalized Iterative Scaling (GIS)to maximize the conditional probability of the training set given the model. MaxEnt tagger has 3.4% error rate. Our investigation examines how much lexical information can be recovered from sublexical features. In order to address these issues we reuse the feature set of MaxEnt in a new model, which we subsequently minimize with the help of linguistically motivated vocabularies.

2 POS tagging Bayesian net

Our tagger combines the features suggested in the literature to date into a Dynamic Bayesian Network (DBN). We briefly introduce the essential aspects of DBNs here and refer the reader to a recent dissertation(Murphy 2002) for an excellent survey. A DBN is a Bayesian network unwrapped in time, such that it can represent dependencies between variables at adjacent time slices. More formally, a DBN consists of two models B^0 and B^+, where B^0 defines the initial distribution over the variables at time 0, by specifying:

- set of variables $X_1, \ldots, X_n$;
- directed acyclic graph over the variables;
- for each variable X_i a table specifying the conditional probability of X_i given its parents in the graph $Pr(X_i|Par\{X_i\})$.

The joint probability distribution over the initial state is:

$$\Pr(X_1, ..., X_n) = \prod_{1}^{n} \Pr(X_i|Par\{X_i\})$$

The transition model B^+ specifies the conditional probability distribution (CPD) over the state at time t given the state at time $t-1$. B^+ consists of:

- directed acyclic graph over the variables $X_1, \ldots, X_n$ and their predecessors $X_1^-, \ldots, X_n^-$ — roots of this graph;
- conditional probability tables $\Pr(X_i|Par\{X_i\})$ for all X_i (but not X_i^-).

The transition probability distribution is:

$$\Pr(X_1, ..., X_n \Big| X_1^-, ..., X_n^-) = \prod_1^n \Pr(X_i | Par\{X_i\})$$

Between them, B^0 and B^+ define a probability distribution over the realizations of a system through time, which justifies calling these BNs "dynamic". In our setting, the word's index in a sentence corresponds to time, while realizations of a system correspond to correctly tagged English sentences. Probabilistic reasoning about such system constitutes inference.

Standard inference algorithms for DBNs are similar to those for HMMs. Note that, while the kind of DBN we consider could be converted into an equivalent HMM, that would render the inference intractable due to a huge resulting state space. In a DBN, some of the variables will typically be observed, while others will be hidden. The typical inference task is to determine the probability distribution over the states of a hidden variable over time, given time series data of the observed variables. This is usually accomplished using the forwardbackward algorithm. Alternatively, we might obtain the most likely sequence of hidden variables using the Viterbi algorithm. These two kinds of inference yield resulting POS tags. Note that there is no need to use "beam search" (cf. Brants 2000).

Learning the parameters of a DBN from data is generally accomplished using the EM algorithm. However, in our model, learning is equivalent to collecting statistics over cooccurrences of feature values and tags. This is implemented in GAWK scripts and takes minutes on the WSJ training corpus. Compare this to GIS or IIS (Improved Iterative Scaling) used by MaxEnt. In large DBNs, exact inference algorithms are intractable, and so a variety of approximate methods has been developed. However, as we explain below, the number of hidden state variables in our model is small enough to allow exact algorithms to work. For the inference we use the standard algorithms, as implemented in the Bayesian network toolkit (BNT, see Murphy 2002).

We base our original DBN on the feature set of Ratnaparkhi's MaxEnt: the set of observable nodes in our network consists of the current word W_i, a set of binary variables C_i, H_i and N_i (for Capitalization, Hyphen and Number) and multivalued variables P_i and S_i (for Prefix and Suffix), where the subscript i stands for position index. There are two hidden variables: T_i and M_i (POS and Memory). Memory represents contextual information about the antepenultimate POS tag. A special value of Memory ("Start") indicates the beginning of the sentence. The POS values are 45 tags of the Penn Treebank tag set (Marcus, Kim, Marcinkiewicz, MacIntyre, Bies, Ferguson, Katz & Schasberger 1994).

Figure 1 represents dependencies among the variables. Clearly, this model makes a few unrealistic assumptions about variable independence and Markov

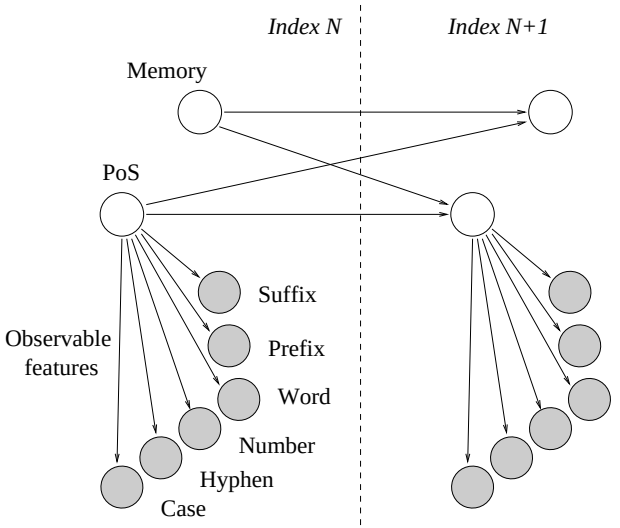

Figure 1: DBN *for PoS tagging.*

property of the sequence. Empirically this does not present a problem. For the discussion of these issues please see Bilmes (2003) who is using similar models for speech recognition. Thus, probability of a complete sequence of POS tags $T_1 \ldots T_n$ is modeled as:

$$\begin{aligned}\Pr(T_1 \ldots T_n) = {} & \Pr(T_1) \times \Pr(F_1|T_1) \times \Pr(T_2|T_1, Start) \times \Pr(F_2|T_2) \times \Pr(M_2|T_1) \\ & \times \textstyle\prod_{i=3}^{n-1} \Pr(T_i|T_{i-1}, M_{i-1}) \times \Pr(M_i|T_{i-1}, M_{i-1}) \times \Pr(F_i|T_i) \\ & \times \Pr(T_n|T_{n-1}, M_{n-1}) \times \Pr(F_n|T_n),\end{aligned}$$

where F_i is a set of features at index $i \in [1..n]$ and:

$$\Pr(F_i|T_i) = \Pr(S_i|T_i) \times \Pr(P_i|T_i) \times \Pr(W_i|T_i) \times \Pr(C_i|T_i) \times \Pr(H_i|T_i) \times \Pr(N_i|T_i)$$

These probabilities are directly estimated from the training corpus.

3 Experiments and Results

We use sections 0-22 of WSJ for training and sections 23, 24 as a final test set. The same split of the data was used in recent publications (Toutanova & Manning 2002; Lafferty, McCallum & Pereira 2001) that report relatively high performance on out-of-vocabulary (OoV) items. The test sections contain 4,792 sentences out of about 55,600 total sentences in WSJ corpus. The average length of a sentence is 23 tokens. The Brown corpus is another part of UPenn TreeBank

dataset, which is of a similar size to WSJ (1,016,277 tokens) but quite different in style and nature. The Brown corpus has substantially richer lexicon and was chosen by us to test the performance on novel text.

We begin our experiments by combining the original MaxEnt feature set into a DBN we call LegoTag to emphasize its compositional nature. The initial network achieves 3.6% error (see Table 1) and closely matches that of MaxEnt (3.4%). Our first step is to reduce the complexity of our tagger because per-

Memory (of values)	**Features**	**Error**		
		Ave	**OoV**	**Sen**
Clustered (5)	Unfactored	4.4	13.0	58.5
Clustered (5)	Factored	3.9	10.8	55.8
Full (45)	Factored	3.6	9.4	51.7

Table 1: *Results for Full LegoTag on WSJ*

forming inference on the DBN containing a conditional probability table of 45^3 elements for Memory variable is cumbersome. At the cost of minor deterioration in performance (3.9%, see Table 1), we compress the representation by clustering Memory values that predict similar distribution over Current tag values. The clustering method is based on Euclidian distance between 45^2 dimensional probability vectors $\Pr(T_i|T_{i-1})$. We perform agglomerative clustering, minimizing the sparseness of clusters (by assigning a given point to the cluster whose farthest point it is closest to). As a result of clustering, the number of Memory values is reduced nine times. Consequently, the conditional probability table of Memory and POS become manageable.

As a second step to simplification of the network, we eliminate feature redundancy. We leave only the lowercase form of each word, prefix and suffix in the respective vocabulary; remove numbers and hyphens from the vocabulary, and use prefix, suffix and hyphen information only if the token is not in the lexicon. The size of the factored vocabularies for word, prefix and suffix is 5705, 2232 and 2420 respectively (a reduction of 12%, 38% and 17%). Comparing the performance of LegoTag with factored and unfactored features clearly indicates that factoring pays off (Table 1). Factored LegoTag is better on unknown words and at the sentence level, as well as overall. In addition, factoring simplifies the tagger by reducing the number of feature values. We report three kinds of results: overall error, error on unknown words (OoV), and per sentence error . Our first result (Table 2) shows the performance of our network without the variable Word. Even when all words in the text are unknown, sublexical features carry enough information to tag almost 89% of the corpus. This success rate is essentially the same as the success rate of the unigram tagger discussed by Charniak, Hendrickson, Jacobson & Perkowitz (1993). However, note that the unigram

Type of LegoTag						Error		
H	**N**	**C**	**P**	**S**	**W**	**Ave**	**OoV**	**Sen**
+	+	+	+	+	−	11.3	11.3	84.0
−	−	+	−	−	+	6.1	30.6	69.0
−	−	−	−	−	+	9.3	47.6	77.7

Table 2: *Results of de-lexicalized and fully lexicalized LegoTag for WSJ*

tagger requires full knowledge of the vocabulary, which is unrealistic in most situations.

To demonstrate this, we test two degenerate variants: one which relies only on lexical information, and another which relies on lexical information plus capitalization. Lexical information alone does very poorly on unknown words, which comes to show that context alone is not enough to uncover the correct POS.

We now turn to the issue of using the morphological cues in POS tagging and create a linguistically "smart" network (Smart LegoTag), whose vocabularies contain a collection of function words, and linguistically relevant prefixes and suffixes assembled from preparatory materials for the English language section of college entrance examination (Scholastic Aptitude Test). For example, the function word vocabulary contains all forms of the verb "be", all question markers (wh-words), modals, prepositions and some non-derived adverbs (like "there"). The vocabularies are very small: 315, 100, and 72, respectively. The percentage of unknown words depends on vocabulary size. For the large lexicon of LegoTag it is less than 12%, while for the Smart LegoTag (whose lexicon contains only function words which are few but very frequent), it is around 50%. In addition, two hybrid networks are created by crossing the suffix set and word lexicon of the Full LegoTag and Smart LegoTag.

The results for the Smart LegoTag, as well as for the Hybrid LegoTags are presented in Table 3. They suggest that nonlexical information is sufficient to assure a stable, albeit not stellar, performance across corpora. Smart LegoTag was trained on WSJ and tested on both WSJ and Brown corpora with very similar results. The sentence accuracy is generally lower for the Brown corpus than for the WSJ corpus, due to the difference in average length. The Hybrid LegoTag with big suffix set and small word lexicon was a little improvement over Smart LegoTag alone. Notably, however, it is better on unknown words than Full LegoTag on the Brown corpus.

The best performance across corpora was registered by the second Hybrid LegoTag (with big word lexicon and small suffix set). This is a very interesting result indicating that the nonlinguistically relevant suffixes in the big lexicon contain a lot of idiosyncratic information about the WSJ corpus and are harmful to performance on different corpora. Full LegoTag and Smart LegoTag encounter qualitatively similar difficulties. Since function words are part of the lexicon of

LegoTag Ftrs		Error						Unkn Wrd %	Unkn Wrd %
		WSJ			Brown				
Word	Suff	Ave	OoV	Sen	Ave	OoV	Sen	WSJ	Brown
5705	2420	3.9	10.0	55.4	10.1	23.4	67.9	11.6	15.4
5705	72	4.4	14.0	58.7	7.7	21.9	69.3		
315	2420	6.4	10.5	70.3	10.1	17.8	76.7	49.2	40.8
315	72	9.6	17.1	82.2	11.4	22.3	82.9		

Table 3: *Results for Smart and Hybrid LegoTags*

both networks, there is no significant change in the success rate over function words. The biggest source of error for both is the noun/adjective (NN/JJ) pair (19.3% of the total error for Smart LegoTag; 21.4% of the total error for Full LegoTag). By and large, both networks accurately classify the proper nouns, while mislabeling adverbs as prepositions and vice versa. The latter mistake is probably due to inconsistency within the corpus (see Ratnaparkhi 1996 for discussion). One place where the two networks differ qualitatively is in their treatment of verbs. Smart LegoTag often mistakes bare verb forms for nouns. This is likely due to the fact that a phrase involving "to" and a following word can be interpreted either analogously to "to mom" (to + NN) or analogously to "to go" (to + VB) in the absence of lexical information. Similar types of contexts could account for the overall increased number of confusions of verb forms with nouns with Smart LegoTag. On the other hand, Smart LegoTag is much better at separating bare verb forms (VB) from present tense verbs (VBP) because it does not rely on lexical information that is potentially confusing since both forms are identical. However, it often fails to differentiate present verbs (VBP) from past tense verbs (VBD), presumably because it does not recognize frequent irregular forms. Adding irregular verbs to the lexicon may be a way of improving Smart LegoTag.

4 Conclusion

DBNs provide an elegant solution to POS tagging. They allow flexibility in selecting the relevant features, representing dependencies and reducing the number of parameters in a principled way. Our experiments with a DBN tagger underline the importance of selecting an efficient feature set. The results clearly show that eliminating redundancies in the feature vocabularies improves performance. Furthermore, while maximal lexicalization can lead to a better score on one corpus, it leads to overfitting. Our solution is to reduce lexicalization by relying on sublexical features. This stratey leads to comparable performance on one corpus, while mainting a higher capacity for generalization. Delexicalized taggers make fewer errors on unknown words, which naturally results in more robust success rate across corpora.

The relevance of a given feature to POS tagging varies across languages. For example, languages with rich morphology would call for a greater reliance on suffix/prefix information. Spelling conventions may increase the role of the Capitalization feature (e.g., German). In the future, we hope to develop methods for automatic induction of efficient feature sets and adapt the DBN tagger to other languages.

REFERENCES

Bilmes, Jeffrey. 2003. "Graphical Models and Automatic Speech Recognition". *Mathematical Foundations of Speech and Language Processing*, (= *IMA Volumes in Mathematics and Its Applications*, 138) ed. by M. Johnson, S. Khudanpur, M. Ostendorf & R. Rosenfeld: 191-246. New York: Springer-Verlag.

Brants, Thörsten. 2000. "TnT a Statistical Part-of-Speech Tagger". *Proceedings of the 6th Conference on Advances in Natural Language Processing*, Seattle, Washington, U.S.A.

Brill, Eric. 1994. "Some Advances In Rule-Based Part-of-Speech Tagging". *Proceedings of the 12th American Association for Artificial Inteligence* (*AAAI-94*), 722-727. Seattle, Washington, U.S.A.

Charniak Eugene, Curtis Hendrickson, Neil Jacobson & Mike Perkowitz. 1993. "Equations for Part-of-Speech Tagging". *Proceedings of the 11th American Association for Artificial Inteligence* (*AAAI-93*), 784-789. Washington, D.C., U.S.A.

Jelinek, Fredereic. 1985. "Markov Source Modeling of Text Generation". *Impact of Processing Techniques on Communication* ed. by J. Skwirzinski. Dordech, Netherlands: Nijhoff.

Kupiec, Julien. 1992. "Robust Part-of-Speech Tagging Using a Hidden Markov Model". *Computer Speech and Language* 6:3.225-242.

Lafferty, John, Andrew McCallum & Fernando Pereira. 2001. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". *Proceedings of the 18th International Conference on Machine Learning*, 282-289. Williamstown, Mass., U.S.A.

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz & Britta Schasberger. 1994. "The Penn Treebank: Annotating Predicate Argument structure". *Proceedings of the ARPA workshop on Human Language Technology*, Princeton, N.J., U.S.A.

Manning, Christopher & Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, Mass.: MIT Press.

Murphy, Kevin. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. dissertation, Univ. of Calif. at Berkeley. Berkeley, Calif.

Ratnaparkhi, Adwait. 1996. “A Maximum Entropy Model for Part-of-Speech Tagging”. *Proceedings of the Empirical Methods in Natural Language Processing Conference* (*EMNLP-96*), 133-142. Philadelphia, Pennsylvania.

Samuelsson, Christer. 1993. “Morphological Tagging Based Entirely on Bayesian Inference”. *Proceedings of the 9th Nordic Conference on Computational Linguistics*, 225-238. Stockholm, Sweden.

Toutanova, Kristina & Christopher Manning. 2000. “Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger”. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora* (*EMNLP-2000*), 63-70. Hong Kong.

Accurate Annotation: An Efficiency Metric

ANTÓNIO BRANCO & JOÃO SILVA
University of Lisbon

Abstract

When accurately tagging corpora for training and testing automatic POS taggers, research issues tend to focus on how to ensure the integrity and consistency of the data produced. The issues we address here are instead: Provided that integrity and consistency are guaranteed, how to evaluate the complexity and efficiency of different accurate tagging procedures?

1 Introduction

While for automatic tagging, both issues of efficiency and accuracy have been addressed, for hand tagging, in turn, research tends to focus around the issue of accuracy (Voutilainen 1999). This is understandable as it is full accuracy that is sought by the hand tagging as this is its specific responsibility in the production cycle of automatic taggers. However, efficiency is definitely also a relevant issue here: provided integrity and consistency of the data can be ensured, what is the most efficient way — requiring the least amount of time — to annotate a corpus with full accuracy?

This question is not only relevant in itself but also because there is a growing perception that the way to obtain better automatic taggers is to have larger training corpora than the ones that have been used so far. Results by Banko & Brill (2001) suggest that the learning curve for machine learning techniques used to extract linguistic knowledge of various kinds, including for POS tagging, is log-linear at least up to 1 billion tokens of training size. Moreover, for the specific task studied in that paper, the best performing algorithm when the training size is 1 million tokens turns out to be the worst performing one when the training size is increased by 3 orders of magnitude. These authors underline that this "suggests that it may make sense for the field to concentrate considerably more effort into enlarging our training corpora and addressing scalability issues, rather than continuing to explore different learning methods applied to the relatively small extent training corpora" (p.1).

As the motto becomes "the larger the better", even if it will be possible to design better techniques to compress training data, the need for massive hand labelling not only does not disappear but increases given there seems to exist no limit for the desirable increase of the size of accurately labelled corpora and consequently for the increase of annotation labor.

The question about the efficiency of accurate tagging is thus relevant. To seek an answer for it, one needs a sensible methodology to objectively estimate

the complexity of the manual tagging of a corpus; and to evaluate and rank major procedures followed for accurately tagging corpora according to their complexity.

2 An objective metric for annotation efficiency

An objective metric for tagging complexity has to be designed in to allow to determine: With respect to a given corpus and two different annotation procedures, which one is the most efficient; And with respect to a given annotation strategy and two corpora (possibly at different stages of development), which corpus implies the largest annotation complexity.

Such a metric has to measure what is essential for the complexity of the annotation task and abstract away from contingent aspects. It has also to be predictive in the sense that it should not need full or partial completion of an annotating task for the complexity of this task to be known beforehand.

Asking a human, or a group of humans, to tag a portion of a corpus and then registering the amount of time consumed by them will not do. Not only this fails to support a predictive viewpoint, as it does not abstract away from nonessential aspects: Differently skilled annotators will take different amounts of time to complete a task of the same size; different ergonomic environments will induce different working rhythms, etc.

Abstracting away from contingent aspects, what remains as the essential dimension in the complexity of tagging is the number of decision steps required. The number of decision steps to annotate a corpus is proportional to the number of tokens that exist in that corpus to be annotated. It is also proportional to the number of tags in the tag set, that is to the number of tags that have to be excluded to determine the tag to be assigned (i.e., the size of the search space to find the correct tag).

If stripped away from contingent details, hand annotation can be conceptualised as the procedure performed by an agent using the following facilitating tool: When in a given point of a corpus, the tool jumps to the next token to be tagged, skipping over the tokens that need not be tagged; after a token is selected, the agent scans through a list of tags (e.g., in a drop down menu) and choose which one to assign. To abstract from possible differences of the agents and isolate the intrinsic complexity of the procedure, what is measured is not the time consumed but the number of steps necessary to complete the assignment task.

Accordingly, to determine the complexity of hand tagging from scratch a corpus, one just needs to know the size of the corpus and of the tag set. If the corpus has N tokens and the tag set T tags, for each token, $(T + 1)/2$ decision steps are needed on average to find the tag to be assigned.[1] Hence, the complexity

[1] Under the assumption that the tags are equiprobable: More on this below.

of accurately tagging that corpus is proportional to $N[(T + 1)/2]$ of what we termed standard tagging steps (sts).

Two important comments are in order here before proceeding. First, this metric is not aimed at providing a measure to determine the complexity involved in annotating a specific token in a specific occurrence: There is no pretension that the abstract procedure described above has any kind of psychological reality. Second, this metric is not aimed either at determining an absolute measure of the complexity of annotating a specific corpus: There is no pretension that the best device to support the hand annotation of a corpus should be one that permits picking one tag in a drop down menu, as a simple voice command might well be envisaged as a better alternative. As we hope to make clear in the discussion below, this metric is to be used as an objective basis to establish, in general terms, a comparative assessment of the average efficiency of different methods that might be conceived to accurately hand annotate a corpus with POS tags.

3 Accurate tagging

Turning now to the identification of the major conceivable procedures to annotate a corpus, we will be assuming that the preparatory preliminaries had been accomplished, the tag set was defined, the facilitation tool is in place, etc.

3.1 *From-scratch method.* The baseline strategy to accurately tag a corpus consists simply in hand annotating it token by token with the appropriate tags.

3.2 *Train-tag-review method.* On a par with this strategy, a method has been advocated based on a "bootstrapping" approach. As far as we were able to trace, its first proposition in a published paper was due to (Day et al. 1997).

This procedure consists in hand tagging a portion of the corpus and using the outcome as seed data to train a first tagger. This tagger is used to tag another stretch of text. The annotation produced is reviewed. The two portions of text are put together and used to train a second tagger, with better accuracy than the first. This cycle can be iterated, with increasingly larger portions accurately tagged and increasingly accurate automatic taggers.

3.3 *Sieve method.* Another "bootstrapping" procedure of a different kind, based in a two-step strategy, can yet be conceived. In the second step, the human annotator just tags the portion of the corpus whose tagging was not terminated in the first step. The first step is accomplished by a "sieving" tool. This tool reduces both the number of tokens that are left to be tagged and the size of the list of admissible tags to be assigned by the human annotator in the second step.

This procedure explores regularities concerning so-called closed and open classes of words.

Closed classes: The few hundreds lexical items of closed classes exhibit very high frequencies. With the help of library reference grammars and online dictionaries: (i) collect the list of those items not pertaining to the classes of Common

Nouns, Adjectives, Verbs, Adverbs ending in *-ly*, Proper Names;[2] (ii) associate each such item with the exhaustive list of its admissible tags (either from closed or open classes); (iii) assemble a sieving tool that by simple lexical lookup, when run over a corpus, tags the tokens of each type collected in (i) with their admissible tag(s).

Open classes: Many items from open classes result from a few derivational processes. As a rule, these processes and the categories of the resulting words can be identified from the word endings. With the help of reference grammars and online dictionaries: (i) collect a list of those word endings and corresponding categories; (ii) for each word ending, collect the corresponding exceptions, i.e., words with that ending but with a category not resulting from the corresponding morphological process;[3] (iii) extend the sieving tool so that after it has tagged closed class tokens, it detects tokens from open classes that bear endings like those in (i) or are one of the exceptions in (ii), and assigns them the admissible tag(s).[4]

This sieving/tagging tool designed along these lines performs the first, sieving step. In the second step, tokens tagged with only one tag are skipped by the human anotator as they are already accurately tagged. To annotate tokens that receive more than one tag, it is enough to pick one of the tags previously assigned. For tagging tokens with no tag yet, it is enough to pick one of the few tags of the open classes left: As adverbs ending in *-ly* were dealt with by the sieving tool, there will be four tags to opt for — Common Nouns, Adjectives, Verbs and Proper Names.[5]

4 Efficiency

In order to check the effectiveness of the metric for tagging complexity, it is now used over each tagging procedure just described. For the sake of having a concrete basis for discussion, let us assume that our task was to accurately tag a corpus with, say, 1 million tokens with an average sized tag set with 39 tags (cf. Annex). In what follows, it will become apparent that the comparative results arrived at would not be affected in case the discussion example opted for was different.

[2] See the tag set used in the Annex

[3] E.g., `ally_CN` is an exception to the rule that assigns `ADV` to tokens ending in *-ly*.

[4] This tool is easily extended to tag also numbers and other tokens that can be described by regular expressions.

[5] Full accuracy is ensured by the fact that if an item was tagged in the first step, it is because it is in the list used by the sieving tool; if it is entered into this list, it is possible and easy to make an exhaustive collection of all its admissible categories. Full coverage of closed classes (or derivationally obtained words from open classes), in turn, is not easy to ensure, but this is harmless: if an item is not entered in the list used by the sieving tool, that item does not receive any tag in the first step and will be found (and accurately tagged) in the second step by the human annotator.

4.1 *From-scratch*

Upper bound. With the from-scratch procedure, we will need to decide which tag to choose for each of the 1 million tokens out of a tag set with 39 tags. If the tags had identical probability of being selected, the annotation of our working corpus would require 20 Msts = $10^6 \cdot [(39+1)/2]$.

Lower bound. However, the tags are not equiprobably selected as different classes of words have different frequencies. The above value for complexity can be reduced if the list of tags presented to the annotator in the drop down menu of the facilitating tool (from which he selects the one to assign) is ranked by the decreasing frequencies of the tags. Hence, a more frequent tag will need less decision steps to be assigned than those required by a less frequent tag (cf. Annex).

To recalculate the complexity value, we take the typical values for the relative frequencies of different classes. This permits to rank the tags and determine how many steps each tag requires to be assigned. In this paper, we use the frequencies of the tags in the Annex observed in an accurately tagged corpus of Portuguese.[6]

The task of annotating, e.g., the Adverbs in our working corpus, will now require $7 \times 0.0529 \times 10^6$ sts as the assignment of tag ADV to each of the 0.0529×10^6 adverbs takes seven steps.

The lower bound for the complexity of our tagging task is obtained by the summation of the similarly computed values for every tag. This amounts to 5 194 129 sts.[7] The complexity of the from-scratch procedure is thus in the range 5.2–20.0 Msts.

4.2 *Train-tag-review*

To estimate the complexity of the train-tag-review procedure, there is a range of options concerning the size of the portion to be used as seed data and the size of each portion to be covered in subsequent tag-review iterations.

Upper bound. *Step 1:* Considering the learning curves in (Brants 2000), for the sake of simplicity, we assume that an initial, 90% accuracy tagger can be obtained with a training corpus of 10 Ktokens. Annotating a text of this size with the facilitating tool with the tags ranked by decreasing frequencies has a complexity of 51 941 sts.[8]

Step 2: To improve the accuracy of the tagger from 90% to 95%, the training corpus is enlarged from 10 to 100 Ktokens. This is done by running the initial tagger over a new portion of 90 Ktokens and then reviewing the outcome. 81

[6] For a discussion of this option, see Section 6. The corpus used has 260 Ktokens accurately tagged with the tag set in the Annex. It contains excerpts from news articles, magazines, and fiction novels. We are grateful to Fernanda Nascimento and Amália Mendes (CLUL) for having granted access to it.

[7] $= 10^6 \times (1 \times 0.1537 + 2 \times 0.1445 + 3 \times 0.1439 + \cdots) = 5\,194\,129.$

[8] $= 10^4 \times (1 \times 0.1537 + 2 \times 0.1445 + 3 \times 0.1439 + \cdots) = 51\,941.$

Ktokens (90% of 90) will be correctly and 9 Ktokens incorrectly tagged. In the reviewing process, each of the 81 Ktokens needs 1 sts to confirm its tag, and each of the remaining 9 Ktokens need as many steps as if it was to be annotated from scratch. For the latter, 46 747 sts will be needed.[9] The annotation of this second portion of 90 Ktokens requires thus 136 747 sts.[10]

Step 3: Let us assume that 97% accuracy can be reached with a corpus of 1 Mtokens. This can be achieved by running the last tagger over a new portion of 900 Ktokens and then reviewing the result. 855 Ktokens (95% of 900) will be correctly and 45K incorrectly tagged. As in step 2, each of the 855 Ktokens requires 1 sts to confirm its tag; each of the 45 Ktokens needs as many steps as if it was tagged from scratch: 233 736 sts. The annotation of this third portion with 900K thus requires 1 088 736 sts.[11]

The task of tagging the working corpus is now complete and the value for its complexity is obtained by summing up the values obtained in each of the above steps. The result is 1 272 230 sts.[12]

Lower bound. A lower bound for the complexity of the tag-train-review procedure is obtained by assuming that we start already with a 98% accuracy tagger, previously developed upon independent training data. This means that the training steps will be bypassed.

After running this tagger over the 1 million corpus, the tags assigned to 98% of it need to be confirmed, while the other 2% are to be reviewed. Each of the 980 Ktokens (98% of 1M) requires 1 sts to confirm its tag. Each of the remaining 20 Ktokens requires as many steps as if it was to be tagged from scratch. This involves 103 883 sts.[13] Taking these values together, the lower bound is determined as 1 083 883 sts.[14]

Considering the figures obtained for the upper and lower bounds concerning the tag-train-review procedure, its complexity is estimated as lying in the range 1.1–1.3 Msts.

4.3 *Sieve*

To estimate the complexity of the new sieve method, we implemented a sieving tool along the lines described in the Section 3.3.[15] A lower bound for the complexity of this method is calculated assuming that this sieving tool has a heuristics to detect Proper Names with 100% precision and recall (on the basis

[9] $= 9 \times 10^3 \times (1 \times 0.1537 + 2 \times 0.1445 + 3 \times 0.1439 + \cdots) = 46\,747.$

[10] $= 90\,000 + 46\,747 = 136\,747.$

[11] $= 855\,000 + 233\,736 = 1\,088\,736.$

[12] $= 46\,747 + 136\,747 + 1\,088\,736 = 1\,272\,230.$

[13] $= 20 \times 10^3 \times (1 \times 0.1537 + 2 \times 0.1445 + \cdots) = 103\,883.$

[14] $= 980\,000 + 103\,883 = 1\,083\,883.$

[15] Following Day et al. (1997), we do not add programming effort to the overall tagging complexity. The sieving tool used in our experiment is now available to be reused for the annotation of other corpora, so in the long run, its implementation cost will be negligible anyway.

of the first letter being capitalized, plus a few rules to handle exceptions). The upper bound, in turn, is calculated by assuming that the sieving tool does not handle Proper Names.

Lower bound. The sieving tool was experimentally run over a corpus with 260 Ktokens ("exp-corpus", from this point, ftn 6). The following results were obtained: No tags: 64%; One tag: 16%; More than one: 20%.

From these values, it can be extrapolated that around 64% of our working corpus of 1 million tokens may be accurately tagged simply by running the sieving tool over it, i.e., without any increase in the complexity of the task of accurately annotating that corpus.

Given this tool is designed to tag a given token iff it assigns to it all its admissible tags, to the other portion of the corpus with tokens that received more than one tag, we refer as the directly detected ambiguity portion (20%). We refer to the portion of the corpus with tokens that received no tag as indirectly detected ambiguity portion (16%).

The tag to assign to each token in the directly detected ambiguity portion is selected from the tags already assigned by the sieving tool. To estimate the complexity of the subsequent disambiguation task, the different degrees of ambiguity involved should be considered. The distribution of tokens with several tags assigned to them is: 2 tags, 59.23%; 3, 29.14%; 4, 0.63%; 5, 0.09%; 6, 1.18%; MWU, 9.34%.

Most of the directly detected ambiguity involves 2 tags per token. As the number of tags per token increases, the frequency of such ambiguities decreases.[16] MWU is a special case of ambiguity where the elements in a sequence of tokens are individually or collectively tagged as a single MWU.

Getting back to the 1 million corpus, the above considerations imply that ca. 200 Ktokens (20%) can be annotated by picking the correct tag from the tags assigned by the sieving tool. For 29.14% of these 200 Ktokens, for instance, this requires picking one tag out of three, thus involving on average 2 sts. Repeating this calculation for each level of ambiguity, the complexity of accurately annotating the directly detected ambiguity portion in the working corpus is estimated as 346 360 sts.[17]

Turning to the indirectly detected ambiguity portion, our experiment shows that after running the sieving tool, ca. 160 Ktokens (16%) receive no tag. Given the tool exhausted the tags to be assigned to closed classes plus Proper Names,

[16] There is an odd increase in the frequency of ambiguities involving 6 tags due to two specific Portuguese forms, viz. como and nada. Both are very frequent and ambiguous: *como* receives the tags `INT`, `REL`, `CJ`, `PREP`, `ADV` and `V`, and *nada* receives `IN`, `DIAG`, `ADV`, `CN`, `ADJ` and `V`.

[17] $= 200 \times 10^3 \times (1.5 \times 0.5923 + 2 \times 0.2914 + 2.5 \times 0.0063 + 3 \times 0.0009 + 3.5 \times 0.0158 + 2 \times 0.0934) = 346\ 360$. We assumed that on average two inspection steps are required to review potential multi-word lexical units.

every token in this portion is potentially ambiguous between three open classes: Adjective, Common Noun or Verb. The task of hand tagging is restricted now to picking one of these three tags.

With the tagged version of the exp-corpus used in the experiment, it is possible to determine that Common nouns are 49.8%, Verbs 36.7% and Adjectives 13.5% of the indirectly detected ambiguity portion. This allows to measure the complexity of the task of annotating the last 160 Ktokens of the working corpus, yet to be tagged. The three possible tags remaining are ranked according to the decreasing order of their frequencies in this portion: Nouns, Verbs, Adjectives. This implies that, for instance, tagging each Adjective requires 2 sts and annotating every Adjective in this indirectly detected ambiguity portion involves 43 200 sts.[18] Putting all the figures together, tagging the last 16% requires 261 920 sts.[19]

Collecting the values for both directly and indirectly detected ambiguity, the lower bound value for the complexity of tagging our working corpus with the sieving procedure is 608 280 sts.[20]

Upper bound. The upper bound is determined by assuming that the sieving tool is not prepared to handle Proper Names. As Proper Names are 7% of the exp-corpus, this implies that the indirectly detected ambiguity portion is enlarged now from 16% to 23% and the tags available to tag this portion are now four: Common Nouns, Adjectives, Verbs and Part of Name.[21] The exp-corpus allows also to know that Common nouns are 34.9%, Part of Names 29.9%, Verbs 25.7%, and Adjectives 9.5% of the indirectly detected ambiguity portion.

It is now possible to obtain the complexity for the indirectly detected ambiguity portion in the working corpus, with 230 Ktokens (23%). It amounts to 482 540 sts.[22]

Note that under this upper bound scenario, the value for the complexity of annotating the directly detected ambiguity portion is the same as the value obtained under in the lower bound scenario: the tokens tagged as Part of Names then are assigned only one tag thus being part of the 64% of the corpus correctly tagged with the sieving tool. Taking thus that value for the complexity of tagging the directly detected ambiguity portion together with the value just calculated above for the indirectly detected ambiguity portion, we obtain 828 900 sts[23] as the upper bound value. Taking the values for the upper and lower bounds of the sieve method, its complexity can be estimated as lying in the range 608–829 Ksts.

[18] $= 160 \times 10^3 \times 2 \times 0.135 = 43\,200$.

[19] $= 160 \times 10^3 \times (1 \times 0.498 + 2 \times 0.367 + 3 \times 0.135)$.

[20] $= 346\,360 + 261\,920 = 608\,280$.

[21] The tokens ambiguous between Proper Names and another tag have a residual value so it can be safely assumed that these 7% were displaced from the portion that received only one tag.

[22] $= 230 \times 10^3 \times (1 \times 0.349 + 2 \times 0.299 + 3 \times 0.257 + 4 \times 0.095)$.

[23] $= 346\,360 + 482\,540 = 828\,900$.

5 Discussion

The values of the upper and lower bounds of tagging complexity for the three procedures are compiled in the table below as well as the efficiency gains by each of them with respect to the others:

			scratch	t-t-r		sieve	
		Ksts	lb	ub	lb	up	lb
scratch	upper	20000	74.03	93.64	94.58	95.85	96.96
	lower	5194		75.51	79.13	84.04	88.29
t-t-r	upper	1272			14.78	34.83	52.20
	lower	1084				23.52	43.90
sieve	upper	829					26.66

When comparing even the best score of the from-scratch procedure with the worst results from the other procedures, the from-scratch method is confirmed as the least efficient one. Any of the other methods permits dramatic savings in terms of hand tagging effort: The train-tag-review allows to save at least 75.51% of the annotation effort, while the sieve method permits to save at least 84.04%.

The results are more instructive when it comes to compare the two more efficient procedures. In the worst case (the train-tag-review procedure's best score is compared with the sieve procedure's worst score), the latter permits to save almost one fourth (23.52%) of the annotation effort needed if the former method is adopted. If the procedures are compared on an equal footing (taking the best scores), the advantage of the sieve method over the train-tag-review one is larger: It permits to save well over one third (43.90%) of the annotation effort.

5.1 Invariance. While the metric for determining tagging complexity is independent of the corpora to be tagged and the human languages in which the data is encoded, the same may not be the case for the complexity scores of the different annotation procedures. Different corpora have different distributions for the frequencies of tags. When using one of the bootstrapping procedures to tag two corpora, it is likely that the two tasks show different complexity values. However, as for each tag, the fluctuation of its relative frequency with respect to different corpora tend to be limited[24] and it is much less than the difference between the complexity of the different methods, the conclusions drawn from the comparison exercise above concerning the ranking of tagging procedures are expected to remain basically valid for a wide range of different corpora.

The same invariance may not hold, however, when different languages are considered, especially if they belong to different language families. When considering generic, large-scale corpora from different languages that can be tagged with the same or approximate tag sets, the relative frequencies of POS tags may present considerable differences. If these differences are large enough to have an

[24] The sieving tool was used over a second corpus with 11.5 million tokens extracted from the corpus available at `http://cgi.portugues.mct.pt/cetempublico/`. The values obtained deviate at most 1 point from the values presented in Section 4.3.

impact on the ranking of the tagging methods according to their complexity now obtained, this is something that has to be empirically verified for each particular case.

5.2 Concluding remarks. While it is important to keep in mind that the ranking of fully accurate tagging procedures may not be independent of the particular language being considered, it is worth noting that the goal here is not to present a definitive ranking of such procedures valid for all languages (something conceivably not possible). Rather, the aim was to show that it is feasible to design an objective, standard metric to predict such ranking when different procedures, different corpora or different languages are taken into account; that the ranking produced is reliable for guiding one to opt for the most efficient procedure; and that the efficiency gains detected (and corresponding gains in terms of labor costs) appeared to be so dramatic that a decision on which procedure to opt for in each case should be considered with the help of such a metric.

Annex

Tag	Category	Freq. (%)	Cplx. (sts)	Tag	Category	Freq. (%)	Cplx. (sts)
CN	common noun	15.37	1	IN	indef. nominal	0.23	21
PNT	punctuation	14.45	2	DFR	fraction denom.	0.21	22
PREP	preposition	14.39	3	ORD	ordinal	0.16	23
V	verb	11.33	4	MTH	month	0.11	24
DA	def. article	11.27	5	WD	week day	0.07	25
PNM	part of name	6.87	6	STT	social title	0.06	26
ADV	adverb	5.29	7	ITJ	interjection	0.06	27
CJ	conjunction	4.77	8	INT	int. pronoun	0.06	28
ADJ	adjective	4.17	9	DIAG	dialogue	0.05	29
PTP	past participle	1.78	10	SYB	symbol	0.05	30
IA	indef. article	1.61	11	EADR	email address	0.03	31
REL	rel. pronoun	1.55	12	PADR	part of address	0.02	32
CL	clitic	1.50	13	MGT	magnitude	0.01	33
DGT	digit	0.86	14	PP	prep. phrase	0.01	34
DEM	demonstrative	0.84	15	DGTR	roman digit	0.00	35
PRS	pers. pronoun	0.71	16	LTR	letter	0.00	36
CARD	cardinal	0.61	17	NP	noun phrase	0.00	37
QD	quant. det.	0.61	18	EOE	end of enum.	0.00	38
POSS	possessive	0.59	19	UNIT	measure unit	0.00	39
GER	gerund	0.30	20				

REFERENCES

Banko, Michele & Eric Brill. 2001. "Scaling to Very Very Large Corpora for Natural Language Disambiguation". *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (*ACL'01*), 26-33. Toulouse, France.

Brants, Thorsten. 2000. "TnT — A Statistical Part-of-speech Tagger". *6th Applied Natural Language Processing Conference*, 224-231. Seattle, Washington.

Day, David, J. Aberdeen, L. Hirschman, R. Kozierok, P. Robinson & M. Vilain. 1997. "Mixed-Initiative Development of Language Processing Systems". *Applied Natural Language Processing Conference* (*ANLP-97*), 348-355. Washington, D.C.

Voutilainen, A. 1999. "An Experiment on the Upper Bound of Interjudge Agreement". *9th Conf. of the European Chapter of the Association for Computational Linguistics* (*EACL'99*), 204-208. Bergen, Norway.

Structured Parameter Estimation for LFG-DOP

MARY HEARNE* & KHALIL SIMA'AN**

* *School of Computing, Dublin City University*
** *ILLC, University of Amsterdam*

Abstract

Despite state-of-the-art performance, the Data Oriented Parsing (DOP) model was shown to suffer from biased parameter estimation, and the good performance seems more the result of ad hoc adjustments than consistent probabilistic generalization. Recently, a new estimation procedure was suggested, called Backoff Estimation, for DOP models based on Phrase-Structure annotations. Backoff Estimation deviates from earlier methods in that it treats the model parameters as a highly structured space of correlated events. In this paper we extend this method towards DOP models based on Lexical-Functional Grammar annotations (i.e., LFG-DOP). We show that the LFG-DOP parameters also constitute a hierarchically structured space. Subsequently, we adapt the Backoff Estimation algorithm from Tree-DOP to LFG-DOP models. Backoff Estimation turns out to be a natural solution to some of the specific problems of robust parsing under LFG-DOP.

1 Introduction

The DOPmodel (Bod 1995, Bod 2001) currently exhibits good performance on impoverished phrase-structure tree-banks. The phrase-structure DOP model, called Tree-DOP, works with the common rewrite operation of substitution (as in Context-Free Grammars). Despite the simple formalism underlying Tree-DOP, probability estimationturns out not as straightforward as it initially seemed. Preceding studies (Johnson 2002, Sima'an & Buratto 2003) have shown the bias of three estimation procedures for Tree-DOP, (Bod 1995, Bonnema et al. 1999, Bod 2001). Recently, Sima'an & Buratto (2003) show that the Tree-DOP parameters cannot be assumed disjoint events, because they abide by a partial order that structures these parameters according to their correlations of occurrence. Sima'an & Buratto (2003) also present an algorithm, Backoff Estimation, that takes into account this fact during parameter estimation for Tree-DOP. Preliminary experiments show improved performance, despite the impoverished first implementation.

In this paper, we study parameter estimation for the extension of DOP to linguistic annotations that are richer than phrase-structure. We concentrate on the extension of DOP to Lexical-Functional Grammar (LFG) annotations, i.e., LFG-DOP (Bod & Kaplan 2003). Naturally, the problem of biasedestimation carries

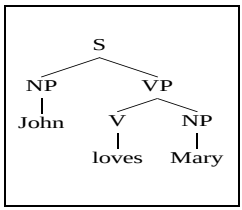

Figure 1: *A toy treebank*

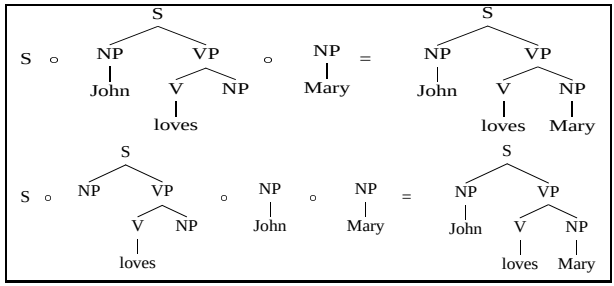

Figure 2: *Two different derivations of the same parse*

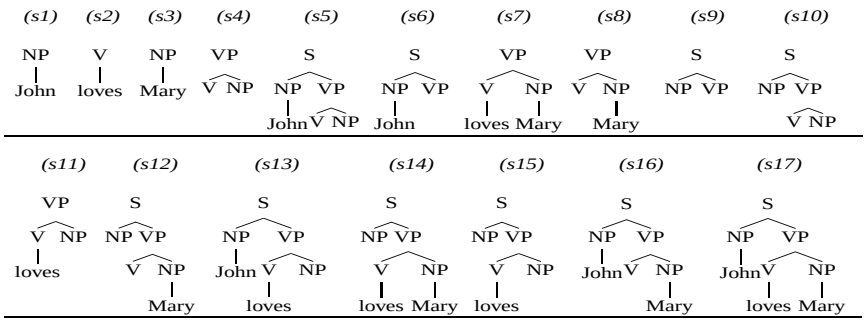

Figure 3: *The subtrees of the treebank in Figure 1*

over from TREE-DOP to LFG-DOP. In fact, the bias in TREE-DOP is further compounded with specific aspects of LFG-DOP that allow for robust processing that can be achieved by abstraction over actually occurring treebank structures. We show how Backoff Estimation applies to LFG-DOP, and how it naturally realizes a specific model architecture that was observed to work best in experiments by (Bod & Kaplan 1998, Bod & Kaplan 2003).

Section 2 provides a review of the TREE-DOP model. Section 3 reviews the Backoff Estimation algorithm. Section 4 reviews LFG-DOP and section 5 extends the Backoff Estimation algorithm to LFG-DOP. Finally, section 6 provides the conclusions from this work.

2 Tree-DOP: Phrase-structure

The Data Oriented Parsing (TREE-DOP) model currently exhibits state-of-the-art performance on benchmark corpora (Rens2001). Like other treebank models, TREE-DOP extracts a finite set of rewrite productions, called *subtrees*, from the training treebank together with probabilities. A connected subgraph of a treebank tree t is called a *subtree* iff it consists of one or more context-free produc-

tions[1] from t. In operational terms, a subtree is determined by selecting in a treebank tree (1) the node that will correspond to the *root* and (2) the nodes that correspond to the *frontier nodes* of that subtree. Following (Bod 1995), the set of rewrite productions of Tree-DOP consists of *all* the subtrees of the treebank trees. Figure 3 exemplifies the set of subtrees extracted from the treebank of Figure 1.

Tree-DOP employs the set of subtrees as a Stochastic Tree-Substitution Grammar (STSG): a TSG is a rewrite system similar to Context-Free Grammars (CFGs); the only difference is that the productions of a TSG are subtrees of arbitrary depth[2]. A TSG derivation proceeds by combining subtrees using the substitution operation $\circ$ starting from the start symbol S of the TSG. In contrast with CFG derivations, multiple TSG derivations may generate the same parse. For example, the parse in Figure 1 can be derived at least in two different ways as shown in Figure 2. In this sense, the Tree-DOP model deviates from other contemporary models, e.g., (Charniak 2000), that belong to the so-called History-Based Stochastic Grammar (HBSG) family (Black et al. 1993). The latter models generate every parse-tree through a unique stochastic derivation.

A Stochastic TSG (STSG) is a TSG extended with a probability mass function P over the set of subtrees: the probability of subtree t, that has root label $\mathtt{R}_t$, is given by $P(t|\mathtt{R}_t)$ i.e., for every non-terminal A: $\sum_{\{t|\mathtt{R}_t=A\}} P(t \mid A) = 1$.

Given a probability function P, the probability of a derivation $D = S \circ t_1 \circ \cdots \circ t_n$ is defined by $P(D \mid S) = \prod_{i=1}^{n} P(t_i|\mathtt{R}_{t_i})$. The probability of a parse is given by the sum of the probabilities of all derivations that generate that parse.

When parsing an input sentence U under a Tree-DOP model, the preferred parse T is the Most Probable Parse (MPP) for that sentence: $\arg\max_T P(T|U)$. However, the problem of computing the MPP is known to be intractable (Sima'an 2002). In contrast, the calculation of the Most Probable Derivation (MPD) D for the input sentence U i.e., $\arg\max_D P(D|U)$, can be done in time cubic in sentence length.

The problem of how to estimate the subtree probabilities from a treebank is *not* as straightforward as originally thought. As shown in (Bonnema et al. 1999, Johnson 2002, Sima'an & Buratto 2003), the common subtree relative-frequency estimator (Bod 1995) is inconsistent and biased in an unintuitive manner. Next, we review a more recent DOP estimator that is based on the Katz backoff smoothing technique (Katz 1987).

[1] Note that a non-leaf node labeled p in tree t dominating a sequence of nodes labeled $c_1, \cdots, c_n$ consists of a graph that represents the context-free production: $p \rightarrow c_1 \cdots c_n$.

[2] Tree depth is the number of edges along the longest path from its root to a leaf node.

3 Backoff Estimation for DOP

We review here the Backoff Estimation procedure presented in (Sima'an & Buratto 2003). The motivation for this estimator comes from the observation that a TREE-DOP model consisting of all subtrees of a given treebank will risk overfitting that treebank. In particular, because the treebank trees themselves are also represented as subtrees, the common Maximum-Likelihood estimator is bound to assign non-zero probability only to the treebank trees – see the example given in (Bonnema et al. 1999). Consequently, (Sima'an & Buratto 2003) apply smoothing to avoid this extreme overfitting problem. The Backoff Estimator is based on the following observation.

Consider the common situation where a subtree[3] t is equal to a tree generated by a derivation $t_1 \circ \cdots \circ t_n$ involving multiple subtrees $t_1 \cdots t_n$. For example, subtree *s17* (Figure 3) can be constructed by different derivations such as *(s16* $\circ$ *s2)*, *(s14* $\circ$ *s1)* and *(s15* $\circ$ *s1* $\circ$ *s3)*. We will refer to subtrees that can be constructed from derivations involving other subtrees with the term *complex subtrees*.

Because the statistics for these different derivations come from the treebank, these statistics must be correlated. For every complex subtree t, we restrict our attention only to the derivations involving pairs of subtrees; in other words, we focus on subtree t such that there exist subtrees t_1 and t_2 such that $t = (t_1 \circ t_2)$. In DOP, the probability of t is given by $P(t|\mathtt{R}_t)$. In contrast, the derivation probability is given by $P(t_1|\mathtt{R}_{t_1})P(t_2|\mathtt{R}_{t_2})$. However, according to the chain rule $P(t_1 \circ t_2|\mathtt{R}_{t_1}) = P(t_1|\mathtt{R}_{t_1})P(t_2|t_1)$. Therefore, the derivation $t_1 \circ t_2$ embodies an independence assumption realized by the approximation[4]: $P(t_2|t_1) \approx P(t_2|\mathtt{R}_{t_2})$. This approximation involves a so-called *backoff*, i.e., a weakening of the conditioning context from $P(t_2|t_1)$ to $P(t_2|\mathtt{R}_{t_2})$. Hence, we will say that the derivation $t_1 \circ t_2$ constitutes *a backoff* of subtree t and we will write ($t \geq_{bfk} t_1 \circ t_2$) to express this fact.

The backoff relation $\geq_{bfk}$ between a subtree and a pair of other subtrees allows for a partial order between the derivations of the subtrees extracted from a treebank. A graphical representation of this partial order is a directed acyclic graph which consists of a node for each pair of subtrees t_i, t_j that constitute a derivation of another complex subtree. A directed edge points from a subtree t_i in a node[5] to another node containing a pair of subtrees $\langle t_j, t_k \rangle$ iff $t_i \geq_{bfk} t_j \circ t_k$. We refer to this graph as the *backoff graph*. An example based on the subtrees of Figure 3 is shown in Figure 4, where $s0$ stands for a subtree consisting of a single node labeled S (the start symbol). We distinguish two sets of subtrees: initial and atomic. Initial subtrees are subtrees that do not participate in a backoff derivation of any other subtree. In Figure 3, subtree $s17$ is the only initial subtree.Atomic

[3] The term "subtree" is reserved for the tree-structures extracted from treebanks.

[4] Note that $\mathtt{R}_{t_2}$ is part of t_1 (the label of the substitution site).

[5] In a pair $\langle t_h, t_i \rangle$ or $\langle t_i, t_h \rangle$ that constitutes a node.

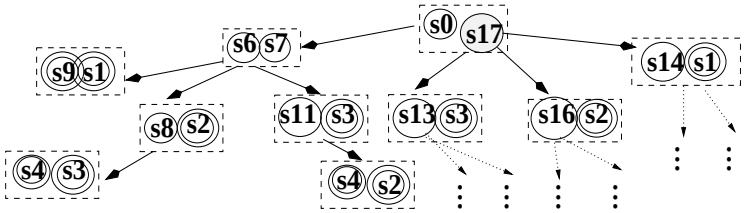

Figure 4: *A portion of the backoff graph for the subtrees in Figure 3*

subtrees are subtrees for which there are no backoffs. In Figure 3, these are subtrees of depth one (double circled in the backoff graph).

In the DOP model (under any known estimation procedure), the probability of a parse-tree is defined as the sum of the probabilities of all derivations that generate this parse-tree. This means that DOP linearly interpolates derivations involving subtrees from different levels of the backoff graph; this is similar to the way Hidden Markov Models interpolate different Markov orders over, e.g., words, for calculating sentence probability. Hence, we will refer to the different levels of subtrees in the backoff graph as the *Markov orders*.

Crucially, the partial order over subtrees (the backoff graph) can be exploited for turning DOP into a "backedoff model" as follows. A subtree is generated by a sequence of derivations *ordered by the backoff relation*. This is in sharp contrast with existing DOP models that consider the different derivations leading to the same subtree as a set of *disjoint* events. Next we present the estimation procedure that accompanies this new realization of DOP as a recursive backoff over the different Markov orders.

Katz Backoff (Chen & Goodman 1998) is a smoothing technique based on the discounting method of Good-Turing (GT) (Chen & Goodman 1998). Given a higher order distribution $P(t|X,Y)$, Katz backoff employs the GT formula for discounting this distribution leading to $P_{GT}(t|X,Y)$. Then, the probability mass that was discounted $(1 - \sum_t P_{GT}(t|X,Y))$ is distributed over the lower order distribution $P(t|X)$.

We assume initial probability estimates P_f based on frequency counts, e.g., as in DOP_{rf} or DOP_{Bon}. The present backoff estimation procedure operates iteratively, top-down over the backoff graph, starting with the initial and moving down to the atomic subtrees, transferring probability mass from complex subtrees to their backoffs.

Let P^c represent the current probability estimate resulting from i previous steps of re-estimation (initially, at step $i = 0$, $P^0 := P_f$). After i steps, the edges of the backoff graph lead to the *current layer* of nodes. For every t, a subtree in a node from the current layer in the backoff graph, an edge e outgoing from t stands for the relation ($t \geq_{bfk} t_1 \circ t_2$), where $\langle t_1, t_2 \rangle$ is the node at the

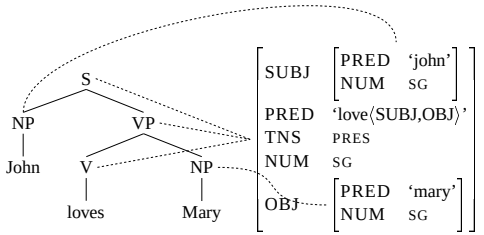

Figure 5: *An LFG representation for 'John loves Mary'*

other end of edge e. We know that $P^c(t|\mathtt{R}_t) = P^c(t_1|\mathtt{R}_{t_1})P^c(t_2|t_1)P^c(t_1 \circ t_2) = P^c(t_1|\mathtt{R}_{t_1})P^c(t_2|\mathtt{R}_{t_2})$. This mean that $P^c(t_2|t_1)$ is backedoff to $P^c(t_2|\mathtt{R}_{t_2})$. Hence, we may adapt the Katz method to estimate the Backoff DOP probability P_{bo} as follows:

$$P_{bo}(t_2|t_1) \;=\; \begin{cases} P^c_{GT}(t_2|t_1) \;+\; \alpha(t_1)\; P_f(t_2|\mathtt{R}_{t_2}) & [\mathtt{P^c(t_2|t_1)} > \mathtt{0}] \\ \alpha(t_1)\; P_f(t_2|\mathtt{R}_{t_2}) & \mathtt{otherwise} \end{cases}$$

where $\alpha(t_1)$ is a normalization factor that guarantees that the sum of the probabilities of subtrees with the same root label is one. Simple arithmetic leads to the following formula: $\alpha(t_1) = 1 - \sum_{t_2 : f(t_1,t_2)>0} P^c_{GT}(t_2|t_1)$.

Using $P_{bo}(t_2|t_1)$, the other backoff estimates are calculated using $P_{bo}(t|\mathtt{R}_t) := P_f(t_1|\mathtt{R}_{t_1})\; P^c_{GT}(t_2|t_1)$ and $P_{bo}(t_1|\mathtt{R}_{t_1}) := (\; 1 + \alpha(t_1)\;)\; P_f(t_1|\mathtt{R}_{t_1})$. Before step $i+1$ takes place over the next layer in the backoff graph, the current probabilities are updated: $P^{i+1}(t_1|\mathtt{R}_{t_1}) := P_{bo}(t_1|\mathtt{R}_{t_1})$.

4 LFG-DOP: Lexical-Functional Grammar

The LFG-DOP model (Bod & Kaplan 1998) employs representations, such as the one in Figure 5, which comprise two parallel levels, constituent structure (c-structure) and functional structure (f-structure), and a mapping between them. The c-structures describe surface structure, the f-structures describe grammatical relations and ϕ maps between these levels of linguistic representation. This relationship is generally expressed as a triple of the form $\langle c, \phi, f\rangle$. An f-structure unit f is ϕ-accessible from a node n if either f is linked to n or f is contained within an f-structure linked to n. This reflects the intuitive idea that nodes can only access information in the units of f-structure to which they are ϕ-linked.

LFG-DOP fragments consist of connected subtrees (as in Tree-DOP) whose nodes are in ϕ correspondence with (partial) f-structures. In operational terminology, the Tree-DOP decomposition of a treebank tree into subtrees proceeds using two operators: *root* and *frontier*. These operators select the nodes that delimit a subtree: *root* selects the root node, and *frontier* selects the frontier nodes. The connected subgraph between these selected nodes constitutes the subtree.

For LFG-DOP, Bod & Kaplan (2003) extend these operators as follows. When a node is selected by the *root operation*, all nodes outside that node's subtree are erased, as in Tree-DOP. Further, all ϕ-links coming from the erased nodes are removed and all f-structure units not ϕ-accessible from the remaining nodes are erased. The *root* operation also deletes the PRED attributes (semantic forms) local to f-structures corresponding to erased nodes. The *frontier* operation selects a set of frontier nodes and deletes all subtrees they dominate, also removing the ϕ-links and semantic forms (but not features) corresponding to any deleted nodes. See example in Figure 6.

LFG-DOP derivations proceed using composition operations that extend the substitution operation of Tree-DOP. The LFG-DOP composition operation ($\circ$) involves two stages: c-structures are combined exactly as in Tree-DOP and their corresponding f-structures are unified recursively.

According to LFG theory (Bresnan 2001), c-structures and f-structures must satisfy certain well-formedness conditions: *uniqueness* specifies that each attribute in the f-structure can have at most one value, *coherence* prohibits the presence of grammatical functions which are not required by the lexical predicate, and *completeness* requires that all functions governed by a lexical predicate must be present in its f-structure. Any parse generated by a sequence of composition operations must satisfy these conditions.

As with Tree-DOP, the probability of a derivation is the product of the probabilities of choosing each of the fragments involved in that derivation $P(f_1 \circ ... \circ f_n) = \prod_i P(f_i)$ and the probability of a parse T is the sum of the probabilities of its distinct derivations $P(T) = \sum_{D\ derives\ T} P(D)$.

In Tree-DOP, the fragment set is partitioned into Competition Sets (CS) according to their root node label. The definition of Competition Sets for LFG-DOP depends on which of the LFG well-formedness conditions are enforced on-line, while the derivation is being constructed, and which are enforced off-line, when the derivation is complete.

In comparison with Tree-DOP, LFG-DOP might suffer from robustness problems because of the constraints expressed in the f-structures (associated with the c-structures). However, this can be addressed by further generalizing fragments generated via *root* and *frontier* as follows.A third operator, *discard*, extracts fragments from f-structures by deleting attribute-value pairs while keeping the associated c-structures and ϕ-links constant (see Figure 7). *Discard* is subject to the restriction that pairs whose values are ϕ-linked to remaining c-structure nodes are not deleted.

Let $|f|$ be the number of occurrences of fragment f in the corpus and CS the competition set of f. According to simple-RF, the probability of f is given by $P(f) = \frac{|f|}{\sum_{f_x : f_x \in CS} |f_x|}$. However *simple-RF* is as biased as fragment relative-frequency estimation for Tree-DOP. Furthermore, it does not account for the fact

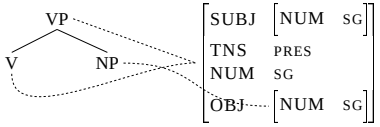

Figure 6: *A fragment generated by* root *and* frontier *from the representation in Figure 5.*

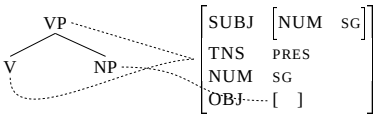

Figure 7: *A fragment generated by the* discard *operation from the representation in Figure 6.*

that *discard*-generated fragments are generalizations of fragments generated by *root* and *frontier*. Because exponentially many fragments can be generated by applying discard to each fragment produced via root and frontier, the discard-generated fragments absorb much of the probability mass under simple-RF estimator, just like larger subtrees absorb a large probability mass in TREE-DOP.

A second estimation method, *discounted-RF*, treats fragments produced by root and frontier as seen events and those generated via discard as unseen events. The relative frequencies of seen events are discounted using Good-Turing and the discounted mass is given to the unseen events. However, this method is as biased as the first. Next we show that the problem of how to employ *discard* within LFG-DOP is merely a symptom of the bias of existing estimates for LFG-DOP as a whole.

5 BackOff Estimation for LFG-DOP

Back-off parameter estimation can be applied to LFG-DOP fragments generated by *root* and *frontier* exactly as described for TREE-DOP, using a directed acyclic graph to represent the partial order between them. A directed edge points from a fragment $\langle c_x, \phi_x, f_x \rangle$ to a pair of fragments $\langle \langle c_y, \phi_y, f_y \rangle, \langle c_z, \phi_z, f_z \rangle \rangle$ if $\langle c_y, \phi_y, f_y \rangle$ and $\langle c_z, \phi_z, f_z \rangle$ compose to give $\langle c_x, \phi_x, f_x \rangle$. Composition of these fragments involves both leftmost substitution over the c-structures and unification over the f-structures.

Because *discard* generated fragments are obtained by generalizing over actual f-structures, the backoff relation in this case is defined in terms of f-structure unification rather than fragment composition. For discard-generated fragments, a directed edge in the graph points from a fragment $\langle c, \phi, f \rangle$ to a pair of fragments $\langle \langle c, \phi, f_y \rangle, \langle c, \phi, f_z \rangle \rangle$ if f-structures f_y and f_z unify to f and $\mathrm{f} \neq \mathrm{f}_y \neq \mathrm{f}_z$: $\langle c, \phi, f \rangle \geq_{bfk} \langle c, \phi, \{f_y \bigcup f_z\} \rangle$.

The probability of the derivation $\langle c, \phi, \{f_y \bigcup f_z\} \rangle$ is given by

$$\begin{aligned} P(\langle c, \phi, \{f_y \bigcup f_z\} \rangle | R_c) &= P(c, \phi | R_c) P(\{f_y \bigcup f_z\} | c, \phi, R_c) \approx \\ P(c, \phi | R_c) P(f_y | c, \phi) P(f_z | c, \phi, f_y) &\approx P(c, \phi | R_c) P(f_y | c, \phi) P(f_z | c, \phi) \end{aligned}$$

This embodies an independence assumption realized by the approximation $P(f_z|c, \phi, f_y) \approx P(f_z|c, \phi)$, which constitutes a backoff; hence the derivation $\langle$c,ϕ,{f$_y \bigcup$ f$_z$}$\rangle$ is said to be a backoff of fragment $\langle$c,ϕ,f$\rangle$. Here, backoff is used to redistribute probability mass among discard-generated LFG-DOP fragments by transferring probability mass from complex fragments to their backoffs in a stepwise manner.

Experiments described in (Bod & Kaplan 2003) suggest that where a parse can be produced without recourse to *discard*-generated fragments, the inclusion of such fragments does not significantly improve parse accuracy. Therefore, it has been suggested that *discard*-generated fragments should only be included in the parse space where the input can be parsed using c-structures only but no parse is possible over fully-instantiated $\langle c, \phi, f \rangle$ fragments.

Way (1999) also observes that *discard* should be used to derive fragments only where absolutely necessary. He suggests that there must be a countable number of cases — such as subject-verb agreement, relative clause agreement and movement phenomena — in which unification fails and *discard*-generated fragments should be applied.

These observations seem to lead towards an LFG-DOP model where *discard*-generated fragments are treated as "second rate" fragments that will be used only upon failure of the other fragments to produce analyzes for the input. We think that the same effect can be realized in a natural manner by Backoff Estimation. Because the parameters are structured in the backoff graph that directs the estimation algorithm, this realizes a kind of *soft*, probabilistic backoff.

6 Conclusions

This paper shows how the parameters for the LFG-DOP model can be estimated as a highly structured space of correlated events. The Backoff Estimation procedure that was originally developed for Tree-DOP (based on Phrase-Structure annotations) turns out especially suitable for LFG-DOP. In particular, Backoff Estimation provides a solution to the problem of robust LFG-DOP parsing without resorting to ad hoc, crisp mechanisms.

Future work will address various aspects of this work. Naturally, future empirical experiments have to exhibit the empirical value of this method. in order to verify the empirical value of this method. There are various specific implementations of the algorithm that need to be sorted out.

REFERENCES

Black, Ezra, Fred Jelinek, John Lafferty, David Magerman, Robert Mercer & Salim Roukos. 1993. "Towards History-based Grammars: Using Richer Models for Probabilistic Parsing". *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics* (*ACL'93*), 31-37. Columbus, Ohio.

Bod, Rens. 1995. *Enriching Linguistics with Statistics: Performance models of Natural Language*. Ph.D. dissertation. ILLC dissertation series 1995-14, University of Amsterdam, The Netherlands.

Bod, Rens & Ron Kaplan. 1998. "A Probabilistic Corpus-Driven Model for Lexical Functional Analysis". *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics* (*ACL-COLING'98*), 145-151. Montréal, Canada.

Bod, Rens. 2001. "What is the Minimal Set of Fragments that Achieves Maximal Parse Accuracy?" *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (*ACL'2001*), 66-73. Toulouse, France.

Bod, Rens & Ron Kaplan. 2003. "A DOP Model for Lexical Functional Grammar". *Data-Oriented Parsing* ed. by R. Bod, R. Scha & K. Sima'an, (= *CSLI Studies in Computational Linguistics*), 211-232. Stanford, Calif.: CSLI Publications.

Bonnema, Remko, Paul Buying & Remko Scha. 1999. "A New Probability Model for Data Oriented Parsing". *Proceedings of the 12th Amsterdam Colloquium* ed. by Paul Dekker, 85-90. Amsterdam: ILLC/Dept. of Philosophy, University of Amsterdam.

Bresnan, Joan. 2001. *Lexical Functional Syntax*. Oxford: Blackwell.

Charniak, Eugene. 2000. "A Maximum Entropy Inspired Parser". *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics* (*NAACL-00*), 132-139. Seattle, Washington, U.S.A.

Chen, Stanley & Joshua Goodman. 1998. "An Empirical Study of Smoothing Techniques for Language Modeling". Technical Report TR-10-98. Boston, Mass.: Harvard University.

Johnson, Mark. 2002. "The DOP Estimation Method is Biased and Inconsistent". *Computational Linguistics* 28:1.71-76.

Katz, Slava. 1987. "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer". *IEEE Transactions on Acoustics, Speech and Signal Processing* (*ASSP*) 35:3.400-401.

Sima'an, Khalil. 2002. "Computational Complexity of Probabilistic Disambiguation". *Grammars* 5:2.125-151.

Sima'an, Khalil & Luciano Buratto. 2003. "Backoff Parameter Estimation for the DOP Model". *Proceedings of the 14th European Conference on Machine Learning (ECML'03), Cavtat-Dubrovnik, Croatia* ed. by N. Lavraĉ et al. (= *Lecture Notes in Artificial Intelligence* (*LNAI*), 2837), 373-384. Berlin: Springer.

Way, Andy 1999. "A Hybrid Architecture for Robust MT using LFG-DOP". *Memory-Based Learning* (Special Issue of *Journal of Experimental and Theoretical Artificial Intelligence*, 11:3), 441-471. London: Taylor & Francis.

Parsing Without Grammar — Using Complete Trees Instead

SANDRA KÜBLER

Seminar für Sprachwissenschaft, University of Tübingen, Germany

Abstract

In recent years, research in parsing has concentrated on weakening the locality of parsing decisions, which led to an increase in the number of parameters to be estimated. This chapter describes a new parsing approach that combines the use of complete trees with an efficient search based on Memory-Based Learning (Stanfill & Waltz 1986, Daelemans, van den Bosch & Zavrel 1999). Parsing is defined as the selection of the most similar syntactic tree in the training data. This tree is then adapted to the input sentence. Concerning function-argument structure, the parser shows a very reliable performance in evaluation (F1=96.52). Errors are very often due to unattached constituents rather than to attachments errors.

1 Introduction

In recent years, research in parsing has concentrated on weakening the locality of parsing decisions (cf. e.g. (Collins 1999, Charniak 2001)), which led to an increase in the number of parameters to be estimated. Many more recent developments are based on the assumption that pure context-free rules, even extended by probabilities, are too local in nature to cover natural language phenomena. *Data-Oriented Parsing* (DOP) (Bod 1998) is the first approach that uses tree fragments of differing size in the search for the best parse tree. However, since the search for the best parse tree cannot be reduced to the search for the best derivation, most DOP models are computationally very intensive.

The problem becomes more severe for non-configurational languages such as German since it is difficult to determine the grammatical function of a constituent based on its local context; the only informative features are the verb itself in combination with morphological and linear information about all the constituents in the sentence. These decisions can only be systematically approached if larger tree fragments are used for parsing. Previous approaches which use Memory-Based Learning (Stanfill & Waltz 1986) for parsing (cf. e.g., (Buchholz 2002, Daelemans, Buchholz & Veenstra 1999, Tjong Kim Sang 2001)), were defined as cascades of classifiers. Such approaches, however, assume a general independence of decisions within one classifier and to a great extent also between levels. These assumptions make such approaches feasible for highly configurational languages such as English, in which the ordering of constituents gives very reliable information about the grammatical function of a constituent. Dagan & Krymolowski (2003), on the other hand, use an adaptation of Memory-Based

Learning: Memory-Based Sequence Learning (MBSL). MBSL is based on the assumption that for learning tasks in the area of shallow parsing, the sequence of wordsis one of the most important types of information. The sequences are represented as tiles including bracketing. In order to parse a sentence, MBSL looks for all covers, i.e. all (potentially overlapping) subsequences of tiles which cover the complete sequence. Then the cover with the highest confidence rating is chosen. By using tiles, MBSL can correctly analyze patterns which do not occur in the training patterns but for which tiles from different training patterns can be put together. Since a high number of tiles needs to be stored for each sentence, it is unfortunately not feasible to lexicalize this approach, which limits its success.

In this chapter, a new parsing strategy, *Memory-Based Parsing* (MBP), that uses complete trees as its repository of grammatical information will be described. Instead of constructing a parse tree from different fragments, MBP defines its task as the search for the most similar complete tree in the instance base. It chooses the most similar tree in an instance base and adapts this tree to the input sentence. The choice of the most similar tree is guided by POS and chunk (cf. Abney 1996) information from preprocessing steps. Previous versions of this system were presented in (Kübler & Hinrichs 2001a, 2001b), a more detailed description of the parser can be found in (Kübler 2002).

2 The Tübingen Treebank of Spoken German, TüBa-D/S

The treebank used for training and evaluation of MBP is the TüBa-D/S treebank (Stegmann et al. 2000). This treebank was developed in the context of VERBMOBIL, a long-term Machine Translation project funded by the German Ministry for Education, Science, Research, and Technology (BMBF). VERBMOBIL had as its goal the speaker-independent translation of spontaneous speech in the domains business appointments, travel scheduling, and hotel reservations. As a consequence, the TüBa-D/S treebank contains transliterations of spontaneous dialogs in these domains. The transliterated data exhibit all the characteristics of spontaneous speech, including hesitation noises, false starts, interruptions, repetitions, and fragmentary utterances.

TüBa-D/S consists of more than 38,000 annotated sentences but since the segmentation onto sentences is rather undetermined for speech data, the sentences in the treebank often consist of more than one sentence in the linguistic sense. Thus, the number of separate trees that were assigned to these sentences reached approx. 67,000 trees. The sentences in TüBa-D/S tend to be rather short (the average sentence length is 4.5 words), the challenge for the parser is rather based on speech phenomena such as ungrammaticality or atypical constituent order than on extremely complex syntactic phenomena. The following sentence, which is taken from the treebank, is typical in this respect: “ich habe

hier übrigens auch schon mir Unterlagen zuschicken lassen von verschiedenen Hotels" (by the way here I have also had brochures sent to me about different hotels already). Here, the modifiers "hier übrigens auch schon" precede the personal pronoun dative object "mir", and the modifier of the accusative object "von verschiedenen Hotels" is extraposed.

For the annotation of the treebank, a surface-oriented annotation scheme has been adopted, which is based on the notion of *topological fields* and enriched by a level of function-argument structure. The linguistic annotations pertain to the levels of morpho-syntax (POS tagging), syntactic phrase structure, and function-argument structure. The annotation relies on a context-free backbone, i.e., proper trees without crossing branches or traces. Long-distance relations are described via specific functional labels.

3 Parsing function-argument structure using an instance base of trees

The parser MBP is based on the assumption that the annotation of the grammatical function for a specific phrase is highly dependent on word order as well as on the other grammatical functions present in the sentence. For this reason, the annotation of grammatical functions must be completed in one step rather than in single independent decisions. However, if the complete syntactic structure is assigned in one step, the parser needs all possible types of information. MBP is designed in a way that it searches for the complete sequence of words, POS tags and chunks.

MBP is divided into two main modules: In the first module, information about the sequences of words and POS tags is used to find the most similar sentence in the instance base. If this module fails to retrieve a reasonably similar sentence, the input sentence is passed to the backing-off module, which relies mainly on chunk information. The latter module is also responsible for the annotation of phrases that were omitted in the first module (cf. below).

3.1 *The search module based on words and POS tags*

Since the selection of the most similar tree for an input sentence needs to be based on the complete sentence rather than on a window of a fixed length (as used in cascaded parsing architectures), the search in the instance base needs to be adapted to the requirement that a flexible number of features serves as the basis for the similarity function. Thus, MBP uses a prefix trie of words as its instance base, and the search for the most similar tree is equivalent to the search for the most similar sequence of words in the trie. Figure 1 shows a sample of such a prefix trie for nine sentences. The final words of the sentences are marked in circles.

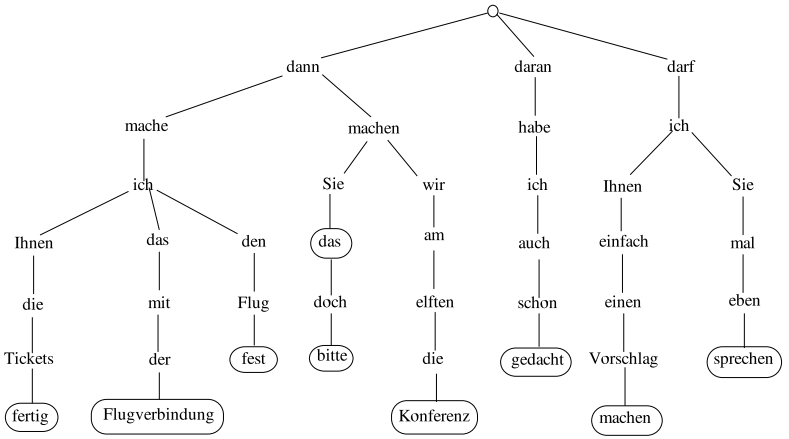

Figure 1: *The prefix trie for nine sentences*

If a word is not found at a specific node in the trie, the search algorithm allows ignoring words or phrases either in the input sentence or in the sentence from the instance base. The phrase boundaries for the input sentence are determined in a pre-processing step via the use of a chunk parser (Abney 1996)[1]. Thus, the search for the input sentence "dann mache ich die Tickets fertig" (then I will arrange the tickets) will retrieve the sentence "dann mache ich Ihnen die Tickets fertig" (then I will arrange the tickets for you). The additional noun phrase in the input sentence, "Ihnen", is omitted in the search. The tree structure that is attached to this sentence is used for assigning a syntactic analysis to the input sentence.

After selecting the most similar sentence in the instance base, the parser is faced with the problem that this sentence generally is not identical to the input sentence. For this reason, the retrieved tree structure needs to be adapted to the input sentence. The parser allows only the most conservative modification: the deletion of the phrases or words that were omitted in the search through the trie. Thus, the structure for the sample input sentence needs to be modified so that the additional noun phrase is removed. The tree structure for the previous

[1] MBP, however, does not only extend the chunk parses with function-argument structure. It rather uses the chunk parse as additional information for the selection of the most similar sentence. There are significant differences in the annotation of phrase boundaries between a chunk parse and a complete syntactic structure due to the deterministic nature of the chunk parser.

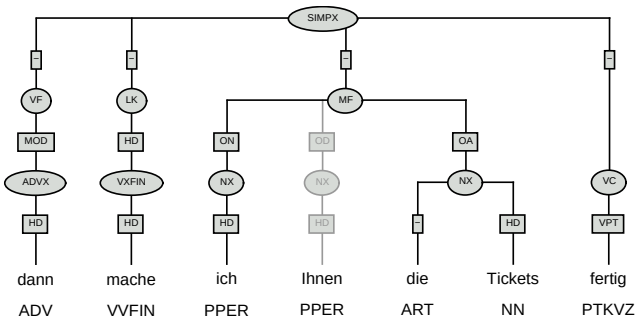

Figure 2: *The most similar tree structure for the input sentence*

example is shown in Figure 2, the phrase to be removed is marked in gray. The sentence is structured into phrases (e.g., NX for "noun phrase" and VXFIN for "finite verb phrase"), topological fields (e.g., VF for "initial field" and LK for "left sentence bracket"), and clauses (SIMPX for "simplex clause"). The edge labels denote head/non-head distinctions within clauses and the grammatical functions of phrases (e.g., ON for "subject", OD for "dative object", OA for "accusative object", and MOD for "underspecified adjunct")[2]. Phrases which could not be attached to the sentence are then looked up separately via the prefix trie (which also contains single phrases, a very typical feature for spontaneous speech). They receive a phrasal annotation, but no attempt is made to attach them to the sentence.

This search strategy as described above is prone to allowing too many phrases or words in the input sentence or in the training sentences to be omitted. In the worst case, the most similar sentence might be selected based on the identity of a single word. Another problem consists in the fact that phrases which are crucial for the structure of the sentence can be omitted. It would be rather unfortunate if the word *Flieger* were omitted in the sentence *es gehen stündlich Flieger ab München* (there go hourly planes from Munich).

These problems are avoided to a high degree by the introduction of cost-based weights. Each word or phrase is assigned a weight (between 0 and 1) which describes the "importance" of the item for the sentence. The weights are calculated during training, based on the relative frequency of the POS tag sequences with and without the item and on the degree to which the tree structure needs to be modified if the item is omitted. Thus, the word *Flieger* will have a higher weight than the phrase *ab München*. At each point where the parser has

[2] For a more thorough introduction to the annotation scheme cf. (Stegmann et al. 2000).

a choice of different actions, it selects the analysis which has accumulated the lowest overall weight, thus executing a best-first search.

3.2 *The backing-off module*

The search strategy described in the previous section is very well suited for finding analyses for sentences that have closely related equivalents in the instance base. Due to the limited size of any treebank, this search strategy will fail for many sentences. In such cases, the parser backs off to a more robust module, which applies less conservative search strategies. The first strategy is to search for longer sentences if the input sentence has been found in the instance base but the last word did not constitute a leaf node. In order to keep this search efficient and to exclude unreasonable analyses, a maximal number of additional words is specified which can be attached to the input sentence. If such a sentence is found, the resulting tree needs to be shortened to fit the input sentence.

If the first strategy fails, the parser attempts to find the sequence of POS tags. Since this search is based on less reliable information that the sequence of words, the omission of words or phrases is not allowed for this search. If a sentence with the same sequence of POS tags is found, the respective tree can be used without adaptation.

The most important strategy in the backing-off module is the search for chunk sequences. For this strategy, the input sentence is first processed by the chunk parser[3]. A chunk parser is a cascade of deterministic finite-state transducers. This allows a very robust and efficient first analysis of the phrasal and clausal structure of a sentence. The input sentence "ab Donnerstag bin ich wieder hier" (from Thursday on I will be here again) is assigned the chunk structure shown in Figure 3.

```
[simpx  [px  [appr ab]
             [nx2  [day Donnerstag]]]
        [fcop bin]
        [nx4  [pper ich]]
        [advx  [adv wieder]]
        [advx  [adv hier]]]
```

Figure 3: *The output of the chunk parser*

The sentences in the instance base are also chunk parsed during the training phase. Based on the similarity between chunk structures concerning yield and

[3] For the present implementation, Steve Abney's chunk parser CASS (Abney 1996) was used.

chunk type, the most similar tree is selected from the instance base. The search can be performed in two different modes: in the first mode, the parser also compares the head words of the chunks, i.e. the search is lexicalized to some extent. If this strategy is not successful, the parser performs the search again without the lexicalization, i.e. only comparing yield and chunk type.

In the above mentioned example, the search for identical chunk sequences will be successful for sentences such as "ab Donnerstag dem dritten bin ich wieder hier" (from Thursday the third on I will be here again) or "nach einer langen Woche sind Sie dann zurück" (after a long week you will be back then). Here, the parser can rely on the chunk analysis, which distinguishes more phrase types than the treebank annotation. The analysis in Figure 3 shows that date phrases, such as "Donnerstag", constitute a different type of noun phrases than pronouns, such as "ich". Both of them are different from standard noun phrases or coordinated noun phrases.

Since the internal structure of the chunks in the input and the most similar sentence may be different, the structure of the most similar sentence needs to be adapted to the input sentence. In order to achieve this, the phrases in the tree structure which correspond to a specific chunk need to be identified and, if they differ in structure from the input sentence, replaced by the correct phrases. Finding correct phrase-internal annotations in the instance base is quite straightforward because phrases with the same POS sequence share the same tree structure in a majority of cases. The identification of phrases corresponding to chunks, however, is far from trivial because a chunk may correspond to more than one phrase. In such cases, several phrases in the tree structure may have to be replaced, and the correct phrases with the correct yield need to be found. In such cases, the POS sequence may be too general: These cases often involve noun chunks which start with one or more adverbs. Here, it is unclear whether the adverbs need to be grouped into a separate adverbial phrase or whether they actually modify the noun phrase. Such a decision generally involves lexical and semantic knowledge, which the parser does not have.

4 Evaluation

MBP is a very efficient parser since it is mostly deterministic. The parser forgoes a complete search of the search space and instead prefers a best-first left-to-right search for the most similar tree. In order to minimize the loss in quality, it uses a wider context than most other parsing algorithms.

MBP was evaluated on the approximately 67,000 trees of TüBa-D/S (cf. Section 2). For every test run, the sentences extracted from the treebank annotations were first sent through preprocessing, i.e., POS tagging using TnT (Brants 2000) and chunk parsing using CASS (Abney 1996). Then MBP assigned complete analyses to the sentences. The time and memory requirements were tested on an

	10-fold CV	leave-one-out
labeled recall (syntactic categories)	82.45%	85.15%
labeled precision (syntactic categories)	87.25%	89.34%
F_1	84.78	87.19
labeled recall (incl. functional categories)	71.72%	76.00%
labeled precision (incl. functional categories)	75.79%	79.65%
F_1	73.70	77.78
functional recall of attached constituents	95.31%	96.56%
functional precision of attached constituents	95.21%	96.48%
F_1	95.26	96.52

Table 1: MBP*'s results for the TüBa-D/S treebank*

Athlon 1700+ (256 MB memory). MBP used approximately 82 MB of memory in training and 52 MB in testing. Training with all 67,000 trees lasts 447 seconds, parsing 6,697 sentence requires 115 seconds, including POS tagging and chunking. In other words, MBP parses more than 58 sentences per second.
The evaluation of MBP was performed with two test designs: *ten-fold cross validation* (10-fold CV) and *leave-one-out testing* (LOO testing). The results of the evaluation are shown in Table 1, in the first column for 10-fold CV, in the second column for LOO testing. The results for 10-fold CV were averaged over the results of the ten test runs, the results for LOO testing over 5,000 test runs. From the parses, the PARSEVAL measures *labeled precision* and *labeled recall*, both based solely on syntactic categories, as well as *(function-) labeled recall* and *(function-) labeled precision*, based on syntactic categories and grammatical functions, were calculated.

The first three metrics concern labeled recall, labeled precision, and the F-score based on only the syntactic categories of the constituents (ignoring the functional labeling) while the next three figures give the performance of the parser when a *combination of syntactic and functional labels* is evaluated. These measures, however, do not make a distinction whether a constituent receives the wrong grammatical function or whether the constituent could not be attached by the parser. In both cases, the combination of syntactic and functional labels for the gold standard and the parser's output are not identical and are thus counted as incorrect. An unattached constituent, however, leads to at least one more incorrect constituent since the node that should dominate this constituent consequently shows the wrong yield. For this reason, the percentage of unattached constituents is also shown, i.e. the percentage of root nodes in the parser output which have an equivalent node in the gold standard, but which is not a root node. These numbers show that a considerable number of constituents could not be attached.

For both types of evaluation, the syntactic evaluation as well as the evaluation on both syntactic and functional labels, the numbers for precision are consider-

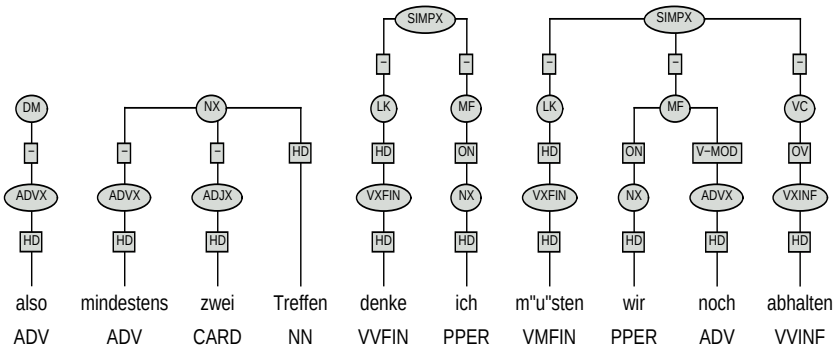

Figure 4: *An example for a partial analysis*

ably higher than for recall. This, in combination with the high percentage of unattached constituents, is an indication that the parser could not find a spanning analysis for sentences in all cases. If, however, such a spanning analysis was found, the parse is extremely reliable.

Additionally, the last three lines in Table 1 give an evaluation of the functional labeling of constituents that could be attached to a higher constituent. Unattached phrases result in low precision numbers since the unattached root node does not contain the correct grammatical function. Figure 4 gives an example for a partial analysis for the sentence "also mindestens zwei Treffen denke ich müßten wir noch abhalten" (so I think we still need to have at least two more meetings).

Here, the parser could not attach the noun phrase "mindestens zwei Treffen" to the remainder of the sentence "müßten wir noch abhalten". When analyzing precision based on syntactic categories and grammatical functions, the root node of the noun phrase would count as incorrect since it does not carry the grammatical function OA, which it would be assigned if it were attached. In the analysis shown in the last three lines in Table 4, consequently only those nodes were counted which are dominated by another node, i.e., non-root nodes. Therefore, the root node of the noun phrase is not considered in the calculation, and for this example, the only incorrect grammatical function is V-MOD, which should correctly be analyzed as MOD. This evaluation based only on attached constituents shows that the parser very reliably recognizes grammatical functions when it is able to find complete analyses and does not have to resort to partial parses. The lower figures in the general evaluation suggest that the parser needs to be extended so that it is more flexible in finding the most similar tree.

The treebank which was used for testing the parser was relatively small as a training set for such a complex task. For this reason, a LOO test was performed to

trie-based search only		backing-off module only	
recall (synt. cat.)	72.02%	recall (synt. cat.)	82.95%
precision (synt. cat.)	77.97%	precision (synt. cat.)	87.96%
F_1	74.88	F_1	85.38
recall (+ func. cat.)	42.65%	recall (+ func. cat.)	70.52%
precision (+ func. cat.)	46.14%	precision (+ func. cat.)	74.73%
F_1	44.33	F_1	72.56
func. recall – attached	97.91%	func. recall – attached	94.63%
func. prec. – attached	97.90%	func. prec. – attached	94.51%

Table 2: *The comparison of tree-based search vs. backing-off*

see how the parsing results change when the parser is provided with more data. It is obvious from the comparison of the two test designs in Table 1 that both precision and recall increased significantly in the LOO tests; this indicates that the performance of the parser would increase even further if more syntactically annotated training data were available.

In order to test the generalization capacity and correctness of the two modules of the parser, each module was tested separately. In the case of the trie search, this means that if the module did not find any similar enough sentence in the instance base, the input sentence was split up into chunks, and the chunks were looked up and annotated, but no attempt was made to group these separate phrases into a tree. In the case of the backing-off module, the sentences were sent directly to this module.

The results of these test runs are shown in Table 2. Looking at the table, it is evident that the figures for the trie-based search are significantly lower than those for the complete parser, which shows that this module has a very restricted coverage. The results for the backing-off module, on the other hand, are slightly higher than for MBP. This is counterbalanced, however, by the lower quality of functional labeling in the cases where the trie-based module was successful in finding a similar sentence in the instance base (cf. rows 4-8 in Table 2). Thus, the combination of both modules gives the best balance between coverage and quality.

5 Conclusion and future work

MBP is a memory-based parser that implements a new approach to parsing: MBP attempts to find the most similar sentence in the instance base relying on many different types of information: the word sequences themselves, their POS tags, and their chunk analysis.

The results of 79.65% correct constituents and grammatical functions in overall trees and of 96.48% correct grammatical functions for attached constituents validate the general approach of a memory-based parser.

Further improvements can be expected from the inclusion of morphological information (cf. (Hinrichs & Trushkina 2002) for such a system) because disambiguated morphological information for phrases is a reliable, and often the only, source for disambiguating the grammatical function of the phrase. Until recently, however, there was no automatic morphological disambiguation available for German.

It is also planned to extend the parser to a k-nn approach similar to the one described in (Streiter 2001), as the use of k nearest neighbors has proven to be advantageous for many MBL applications. Since for memory-based parsing, a higher number of k would lead to several competing tree structures, Streiter's algorithm selects a number of candidates for the most similar sentence and then uses a more fine grained search to select the ultimate candidate. This search can be based on EDIT DISTANCE when the deleting and adding actions are performed on chunks. Such an modification of the algorithm would allow for a more flexible search for the most similar tree since the parser will no longer be restricted to finding complete prefixes of word sequences.

The *leave-one-out* evaluation shows that the size of the training corpus is rather small for finding the most similar sentence in the instance base. Annotating more sentences manually, however, is is rather expensive. But there are methods which would allow an automatic extension of the data: One possibility would be the use of the active-passive diathesis in order to generate the alternates for the sentences in the treebank. Another way of modifying sentences and their trees automatically could be based on the relatively free word order in German. As mentioned above, German only has relatively few restrictions on the order of no-verb phrases in the sentence. Since MBP does not break a tree into rules, it cannot cover all possible phrase orders of a sentence. Thus it would profit if different alternations of a sentence could be provided.

REFERENCES

Abney, Steven. 1996. "Partial Parsing via Finite-state Cascades". *Journal of Natural Language Engineering* 2:4.337-344.

Bod, Rens. 1998. *Beyond Grammar: An Experience-Based Theory of Language* (= *CSLI Lecture Notes*, 88). Stanford, Calif.: CSLI Publications.

Brants, Thorsten. 2000. "TnT — a Statistical Part-of-speech Tagger". *Proceedings of the 6th Conference on Applied Natural Language Processing* (*ANLP-2000*), 224-231. Seattle, Washington, U.S.A.

Buchholz, Sabine. 2002. *Memory-Based Grammatical Relation Finding.* Ph. D. dissertation, University of Tilburg, The Netherlands.

Charniak, Eugene. 2001. "Immediate Head Parsing for Language Models". *Proceedings of the 39th Annual Meeting of the ACL and the 10th Conference of the European Chapter of the ACL*, (*ACL/EACL 2001*), 116-123. Toulouse, France.

Collins, Michael. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. dissertation, University of Pennsylvania.

Daelemans, Walter, Sabine Buchholz & Jorn Veenstra. 1999. "Memory-Based Shallow Parsing". *Proceedings of the Conference on Computational Natural Language Learning* (*CoNLL-99*), 53-60. Bergen, Norway.

Daelemans, Walter, Antal van den Bosch & Jakub Zavrel. 1999. "Forgetting Exceptions is Harmful in Language Learning". *Machine Learning: Special Issue on Natural Language Learning* 34.11-43.

Dagan, Ido & Yuval Krymolowski. 2003. "Compositional Memory-Based Partial Parsing". *Data-Oriented Parsing* ed. by Rens Bod, Remko Scha & Khalil Sima'an, 169-188. Stanford, Calif.: CSLI Publications.

Hinrichs, Erhard W. & Julia S. Trushkina. 2002. "Forging Agreement: Morphological Disambiguation of Noun Phrases". *Proceedings of the First Workshop on Treebanks and Linguistic Theory* (*TLT 2002*) ed. by Erhard Hinrichs & Kiril Simov, 78-95. Sozopol, Bulgaria.

Kübler, Sandra. 2002. *Memory-Based Parsing of a German Corpus*. Ph.D. dissertation, Seminar für Sprachwissenschaft, Universität Tübingen, Germany.

Kübler, Sandra & Erhard W. Hinrichs. 2001a. "From Chunks to Function-Argument Structure: A Similarity-based Approach". *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (*ACL/EACL'01*), 338-345. Toulouse, France.

Kübler, Sandra & Erhard W. Hinrichs. 2001b. "TüSBL: A Similarity-based Chunk Parser for Robust Syntactic Processing". *Proceedings of the First International Human Language Technology Conference* (*HLT-2001*), 356-361. San Diego, Calif., U.S.A.

Stanfill, Craig & David L. Waltz. 1986. "Towards Memory-based Reasoning". *Communications of the ACM* 29:12.1213-1228.

Stegmann, Rosmary, Heike Telljohann & Erhard W. Hinrichs. 2000. "Stylebook for the German Treebank in VERBMOBIL". Verbmobil Technical Report 239, Tübingen, Germany.

Streiter, Oliver. 2001. "Recursive Top-down Fuzzy Match, New Perspectives on Memory-based Parsing". *Proceedings of the 15th Pacific Asia Conference on Language, Information and Computation* (*PACLIC 2001*), 345-356. Hong Kong.

Tjong Kim Sang, Erik F. 2001. "Transforming a Chunker to a Parser". *Computational Linguistics in the Netherlands 2000* ed. by Walter Daelemans, Khalil Sima'an, Jorn Veenstra & Jakub Zavrel, 177-188. Amsterdam: Rodopi.

Phrase Recognition by Filtering and Ranking with Perceptrons

XAVIER CARRERAS & LLUÍS MÀRQUEZ
TALP, LSI, Universitat Politècnica de Catalunya

Abstract

This work presents a phrase recognition system based on perceptrons, and an online learning algorithm to train them together. The recognition strategy applies learning in two layers, first at word level, to filter words and form phrase candidates, second at phrase level, to rank phrases and select the optimal coherent set. We provide a global feedback rule which reflects the dependencies among perceptrons and allows to train them together online. Experimentation on partial parsing problems and named entity extraction gives state-of-the-art results on the CONLL public datasets. We also provide empirical evidence that training the functions together is clearly better than training them separately, as in the conventional approach.

1 Introduction

Over the past few years, many machine learning methods have been applied to NLP tasks in which phrases of some type have to be recognized. Generally, given a sentence — as a sequence of words — the task is to predict a bracketing for the sentence representing a structure of phrases, either sequential (e.g., base syntactic chunks, named entities, etc.) or hierarchical (e.g., clause structure, parse trees, etc.). The usual approach under the discriminative paradigm consists of decomposing the global phrase recognition problem into a number of local learnable subproblems, and infer the global solution from the outcomes of the local subproblems. For chunking the approach is typically to perform a tagging. In this case, local subproblems include learning whether a word *opens*, *closes*, or is *inside* a phrase of some type, and the inference process consists of computing the optimal tag sequence which encodes the phrases, by means of dynamic programming (Punyakanok & Roth 2001, Kudo & Matsumoto 2001). To recognize hierarchical structure, additional local decisions are required to determine the embedding of phrases, resulting in a more complex inference process which recursively builds the global solution (Ratnaparkhi 1999, Carreras et al. 2002).

A usual approach for solving the local subproblems is to learn a separate classifier for each decision, by maximizing some local measure of performance, such as classification accuracy on the local decision. However, when performing the phrase recognition task, the classifiers are used together and dependently. Moreover, the global performance of a system is measured in terms of precision and recall of the recognized phrases, which is not directly the local classification

accuracy for which the classifiers are usually trained. Recent works on the area provide alternative learning strategies in which the learning process is guided from a global point of view. In the online setting, Collins (2002) presents a variant of the perceptron for tagging, in which the learning feedback is globally given from the output of the Viterbi decoding algorithm. Also, Crammer & Singer (2003) present a topic-ranking algorithm, in which several perceptrons receive feedback from the ranking they produce over a training instance. Alternatively, recent works on conditional random fields provide techniques to estimate a global conditional distribution of probabilities which models the task (Sha & Pereira 2003).

In this paper, we present a learning architecture based on filters and rankers for the general task of recognizing phrases in a sentence. Given a sentence, learning is first applied at word level to filter out non plausible phrase candidates. Then, learning is applied at phrase level to score phrase candidates and decide the optimal set to form the solution. The overall strategy can be seen, therefore, as an inference process which, guided by the learned functions, explores only a plausible set of coherent solutions to find the best scored one. It is worth noting that the second layer deals with phrase candidates and may take into account partially constructed phrase structures. An advantage of working at this level is that very rich and informed feature representations can be used to exploit structural properties of the examples, possibly through the use of kernel functions. However, a disadvantage of working with high level constructs is that the number of candidates to explore increases (e.g., there is a quadratic number of phrases with respect to the number of words in a sequence) and the search space may become too large to be explored. This fact motivates the introduction of the word–level filtering layer, which makes the global problem computationally tractable by reducing the search space in which the high–level layer operates.

Our main contribution is a recognition-based feedback rule which allows to learn the decisions in the system as perceptrons, all in one go. The learning strategy works online at sentence level. When visiting a sentence, the perceptrons are first used to recognize the set of phrases, and then updated according to the correctness of the solution. The update rule reflects to each perceptron its committed errors from a global point of view, in a conservative manner. As a result, the learned functions are automatically approximated to behave as word filters and phrase rankers, and thus, become adapted to the recognition strategy.

The evaluation of the presented architecture has been performed using the CoNLL public datasets, obtaining state-of-the-art performance on three relevant NLP problems (namely chunking, clause identification and named entity extraction), while being conceptually simple and flexible. We also provide empirical evidence that training the functions together is clearly better than training them separately, as in the usual approach.

2 The phrase recognition model

Let x be a sentence consisting of a sequence of n words $[x_1, x_2, \ldots, x_n]$, belonging to the sentence space $\mathcal{X}$. Let $\mathcal{K}$ be a finite set of predefined phrase categories. A *phrase*, denoted as $(s, e)_k$, is the sequence of consecutive words spanning from word x_s to word x_e, with $s \leq e$ and $k \in \mathcal{K}$. Let $p_1 = (s_1, e_1)_{k_1}$ and $p_2 = (s_2, e_2)_{k_2}$ be two different phrases. We define that p_1 and p_2 *overlap* iff $s_1 < s_2 \leq e_1 < e_2$ or $s_2 < s_1 \leq e_2 < e_1$, and we note it as $p_1 \sim p_2$. Also, we define that p_1 is *embedded* in p_2 iff $s_2 \leq s_1 \leq e_1 \leq e_2$, and we note it as $p_1 \prec p_2$.

Let $\mathcal{P}$ be the set of all possible phrases, i.e., $\mathcal{P} = \{(s, e)_k \mid 1 \leq s \leq e, k \in \mathcal{K}\}$. In a phrase recognition problem, a *solution* for an input sentence x is a finite set y of phrases which is *coherent* with respect to some *constraints*. We consider two types of constraints: overlapping and embedding. For the problem of recognizing sequentially organized phrases, often referred to as *chunking*, phrases are not allowed to overlap or embed. Thus, the solution space can be formally expressed as $\mathcal{Y} = \{y \subseteq \mathcal{P} \mid \forall\, p_1, p_2 \in y \;\; p_1 \not\sim p_2 \wedge p_1 \not\prec p_2\}$. For the problem of recognizing phrases hierarchically organized, a solution is a set of phrases which do not overlap but may be embedded, $\mathcal{Y} = \{y \subseteq \mathcal{P} \mid \forall\, p_1, p_2 \in y \;\; p_1 \not\sim p_2\}$.

The learning problem for phrase recognition is as follows. Given a training set $S = \{(x^1, y^1), \ldots, (x^m, y^m)\}$, where x^i are sentences in $\mathcal{X}$ and y^i are solutions in $\mathcal{Y}$, the goal is to learn a function $\mathrm{R} : \mathcal{X} \rightarrow \mathcal{Y}$ which correctly recognizes phrases on unseen sentences. We consider two components within this function, both being learning components of the recognizer. First, a *filtering* function F which, given a sentence x, identifies a set of candidate phrases, not necessarily coherent, for the sentence, $\mathrm{F}(x) \subseteq \mathcal{P}$. Second, a *score* function which, given a phrase, produces a real-valued prediction indicating the plausibility of the phrase.

Using the two components, the *phrase recognizer*, R, is modeled as a function which searches a coherent phrase set for a sentence x according to the following optimality criterion:

$$\mathrm{R}(x) = \arg \max_{y \subseteq \mathrm{F}(x) \mid y \in \mathcal{Y}} \sum_{(s,e)_k \in y} \mathrm{score}((s, e)_k, x, y) \tag{1}$$

That is, among all the coherent subsets of candidate phrases, the optimal solution is the one whose phrases maximize the summation of phrase scores.

The filtering function F is used in a first layer to reduce the search space of the function R. It is intended to filter out phrase candidates from $\mathcal{P}$ by applying decisions at word level but without discarding actual phrases in the solution (i.e., trying to maintain recall levels as high as possible). A simple setting for this function is a *start-end* classification for each phrase type: each word of the sentence is considered as k-*start* —if it is likely to start a type-k phrase— and as k-*end* —if it is likely to end a type-k phrase. Each k-*start* word x_s with each

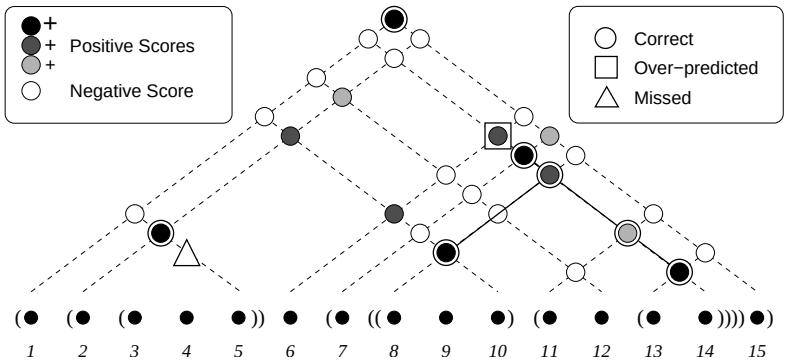

Figure 1: *Example of the strategy for recognizing phrases*

k-*end* word x_e, having $s \leq e$, form the phrase candidate $(s, e)_k$. Alternatives to this setting may be to consider a single pair of *start-end* classifiers, independent of the phrase types, or to do a different tagging for identifying phrases, such as a *begin-inside* classification.

Once the phrase candidates are identified, the optimal coherent phrase set is selected according to (1). There is no need to explicitly enumerate each possible coherent phrase set, which would result in an exponential exploration. Instead, by guiding the exploration through the problem constraints and using dynamic programming the optimal coherent phrase set can be found in polynomial time with respect to the sentence length. For chunking problems, the solution can be found in quadratic time by performing a Viterbi-style exploration from left to right (Punyakanok & Roth 2001). When embedding of phrases is allowed, a cubic-time bottom-up exploration is required (Carreras et al. 2002).

Figure 1 shows a schematic example of how the recognition strategy works, considering generic phrases without type for simplicity. The input sentence is represented at the bottom as a sequence of words $x_1 \ldots x_{15}$, each being a small black circle. The bracketing represents the correct solution for the sentence.

First, the *start-end* functions are applied to each word. An oblique dashed line indicates each positive classification: for *start* words (x_1, x_2, x_6, x_7, x_8, x_{11}, x_{13}) the line goes to the right, whereas for *end* words (x_5, x_{10}, x_{12}, x_{14}, x_{15}) the line goes to the left. Note that these local predictions produce errors, such as the missing start at x_3. The intersections of dashed lines correspond to phrase candidates, printed as circles. For instance, the start word x_7 together with the end word x_{14} forms the phrase candidate $(7, 14)$. Note that in the filtered space there are only 27 phrase candidates out of the 120 possible phrases. The grey scale of circles indicates the prediction of the *score* function to each phrase candidate. White indicates a negative prediction (phrase candidate to be rejected), while

greys indicate three degrees of positive predictions for phrases (confidence increases from light to dark). Sentence processing is performed by levels from the bottom-up, i.e., exploring from short to long phrase candidates, and left-to-right at each level. While predicting phrases, the optimal structure at the explored region is maintained. Thus, the prediction method applied to a phrase can take advantage of the predicted solution found within the phrase. For instance, when scoring the phrase $(7, 14)$ a hierarchy of four phrases is found inside. When representing the phrase into features, high-level patterns of the inside structure can be exploited, possibly by means of kernel functions.

At the end of the process, the global optimal structure has been computed. The selected phrases are marked with circles (for corrects) or squares (for the over-predicted). The selection has been made so as to maximize scores, respecting overlapping constraints in the selected set. A correct phrase, missed in the solution, appears as a triangle. Note that among all local errors produced in the process by the three functions (*start* at words 3 and 6, *end* at word 12, and *score* at phrase candidates $(2, 10)$, $(2, 12)$, $(6, 10)$, $(6, 14)$, and $(7, 15)$), only three are critical at propagating to the global solution: *start* at words 3 and 6, and *score* at $(6, 14)$. The latter are the only ones that produce an update to the classifiers' weights during training. See description of the learning algorithm in Section 3.

3 Online learning via recognition feedback

In this section, we describe an online algorithm for training the learning components of the phrase recognizer, namely the *start-end* classifiers in F and the *score* function. The learning challenge consists in approximating the functions so as to maximize the global F_1 measure on the problem, taking into account that the functions interact. Each function is implemented using a linear separator, $\mathrm{h}_{\mathbf{w}} : \mathbb{R}^n \to \mathbb{R}$, operating in a feature space defined by a feature representation function, $\phi : \mathcal{X} \to \mathbb{R}^n$, for some instance space $\mathcal{X}$. The function F consists of two classifiers per phrase type: the *start* classifier ($\mathrm{h}^k_{\mathrm{S}}$) and the *end* classifier ($\mathrm{h}^k_{\mathrm{E}}$). Thus, the F function is formed by a prediction vector for each classifier, noted as $\mathbf{w}^k_{\mathrm{S}}$ or $\mathbf{w}^k_{\mathrm{E}}$, and a shared representation function ϕ_{w} which maps a word in context into a feature vector. A prediction on a word x_i is computed as $\mathrm{h}^k_{\mathrm{S}}(x_i, x) = \mathbf{w}^k_{\mathrm{S}} \cdot \phi_{\mathrm{w}}(x_i, x)$, and similarly for the $\mathrm{h}^k_{\mathrm{E}}$, and the sign is taken as the binary classification. The *score* function computes a real-valued score for a phrase candidate $(s, e)_k$. We implement this function with a prediction vector $\mathbf{w}^k$ for each type $k \in \mathcal{K}$, and also a shared representation function ϕ_{p} which maps a phrase into a feature vector. The score prediction is then given by the expression: $\mathrm{score}((s, e)_k, x, y) = \mathbf{w}^k \cdot \phi_{\mathrm{p}}((s, e)_k, x, y)$.

The mistake-driven online learning algorithm proposed will be referred to as FR-Perceptron since it is a perceptron-based learning algorithm that approximates the prediction vectors in F as *filters* of words, and the score vectors as

rankers of phrases. It starts with all vectors initialized to 0, and then runs repeatedly in a number of epochs T through all the sentences in the training set. Given a sentence, it predicts the optimal phrase solution given by (1) using the current vectors. As in the perceptron algorithm, if the predicted phrase set is not perfect the vectors responsible of the incorrect prediction are updated additively. The pseudocode is as follows:

- Input: $\{(x^1, y^1), \ldots, (x^m, y^m)\}$, x^i are sentences, y^i are solutions in $\mathcal{Y}$
- Define: $W = \{\mathbf{w}^k_{\mathrm{S}}, \mathbf{w}^k_{\mathrm{E}}, \mathbf{w}^k | k \in \mathcal{K}\}$.
- Initialize: $\forall \mathbf{w} \in W$ $\mathbf{w} = \mathbf{0}$;
- for $t = 1 \ldots T$, for $i = 1 \ldots m$:
  (1) $\hat{y} = \mathrm{R}_W(x^i)$
  (2) $\mathrm{recognition\_learning\_feedback}(W, x^i, y^i, \hat{y})$
- Output: the vectors in W.

We now describe the recognition learning feedback, designed to naturally fit the phrase recognition setting. Let y^* be the gold set of phrases for a sentence x, and $\hat{y}$ the set predicted by the R function. Let $\mathrm{goldS}(x_i, k)$ and $\mathrm{goldE}(x_i, k)$ be, respectively, the perfect indicator functions for *start* and *end* boundaries of phrases. That is, they return 1 if word x_i starts/ends some k-phrase in y^* and -1 otherwise. The algorithm is mistake driven, which means that for all the phrases correctly identified the recognition learning feedback does nothing. Regarding the errors, we differentiate two kinds of phrases in order to give feedback to the functions being learned:

(i) *Missed phrases,* $\forall (s, e)_k \in \ y^* \backslash \hat{y}$:

(1) Update misclassified boundary words:
if $(\mathbf{w}^k_{\mathrm{S}} \cdot \phi_{\mathrm{w}}(x_s) \leq 0)$ then $\mathbf{w}^k_{\mathrm{S}} = \mathbf{w}^k_{\mathrm{S}} + \phi_{\mathrm{w}}(x_s)$
if $(\mathbf{w}^k_{\mathrm{E}} \cdot \phi_{\mathrm{w}}(x_e) \leq 0)$ then $\mathbf{w}^k_{\mathrm{E}} = \mathbf{w}^k_{\mathrm{E}} + \phi_{\mathrm{w}}(x_e)$

(2) Update score function, if applied:
if $(\mathbf{w}^k_{\mathrm{S}} \cdot \phi_{\mathrm{w}}(x_s) > 0 \wedge \mathbf{w}^k_{\mathrm{E}} \cdot \phi_{\mathrm{w}}(x_e) > 0)$ then $\mathbf{w}^k = \mathbf{w}^k + \phi_{\mathrm{p}}((s, e)_k)$

(ii) *Over-predicted phrases,* $\forall (s, e)_k \in \ \hat{y} \backslash y^*$:

(1) Update score function:
$\mathbf{w}^k = \mathbf{w}^k - \phi_{\mathrm{p}}((s, e)_k)$

(2) Update words misclassified as S or E:
if $(\mathrm{goldS}(x_s, k) = -1)$ then $\mathbf{w}^k_{\mathrm{S}} = \mathbf{w}^k_{\mathrm{S}} - \phi_{\mathrm{w}}(x_s)$
if $(\mathrm{goldE}(x_e, k) = -1)$ then $\mathbf{w}^k_{\mathrm{E}} = \mathbf{w}^k_{\mathrm{E}} - \phi_{\mathrm{w}}(x_e)$

As a practical issue, it should be noted that in our implementation all updates are performed in parallel, that is, the conditions of the feedback rule are evaluated with the initial set of vectors. Additionally, the updates of the *start-end* vectors are performed only once per word, although an incorrect prediction on a word may generate several incorrect phrases.

This feedback models the interaction between the two layers of the recognition process. The *start-end* layer filters out phrase candidates for the scoring layer. Thus, misclassifying a boundary word of a correct phrase blocks the generation of the candidate and produces a missed phrase. Therefore, *start* or *end* prediction vectors are moved toward the misclassified boundary words of a missed phrase. When an incorrect phrase is predicted, the vectors of the *start* or *end* words are moved away, provided that they are not actual boundary words in the solution. Note that we do not care about false positives, i.e., *start* or *end* words which do not finally over-produce a phrase. Regarding the scoring layer, each category vector is moved toward missed phrases and away from the over-predicted.

It is important to note that the feedback rule operates only on the basis of the predicted solution $\hat{y}$, avoiding to make updates for every prediction the function has made. Thus, the learning strategy takes advantage of the recognition process, and concentrates on assigning high scores for the correct phrases and making the incorrect competing phrases to score lower than the correct ones. Consequently, it tends to approximate the desired behavior of the global R function, i.e., to make the summation of the scores of the correct phrase set maximal with respect to other phrase set candidates. This learning strategy is closely related to other works on learning ranking functions (Collins 2002; Crammer & Singer 2003, Har-Peled et al. 2003).

4 Phrase recognition in natural language

In this section we briefly describe problems in the natural language domain in which phrases of some type have to be recognized. For all the problems we followed the setting of the CoNLL conference shared tasks, yearly organized by the ACL's Special Interest Group on Natural Language Learning. For more information one may consult `http://cnts.uia.ac.be/signll/shared.html`.

The first two problems concern the recognition of syntactic phrases for the English language, and are released with data of the Penn Treebank. In **chunking**, also known as shallow parsing, the base syntactic phrases, or chunks, of a sentence have to be recognized. The chunks in a sentence can not overlap and are non-recursive, that is, they can not be embedded. The CoNLL-2000 task consisted of recognizing 11 different types of chunks on the basis of words and part-of-speech tags. The second problem, **clause identification**, consists of recognizing the clauses of a sentence. A clause can be roughly defined as a phrase with a subject, possibly implicit, and a predicate. Clauses in a sentence form a hierarchical structure which constitutes the skeleton of the full syntactic tree. Thus, embedding of clauses is allowed. The goal of the CoNLL-2001 task was to recognize clauses on the basis of words, part-of-speech tags and base chunks.

The last problem is **named entity recognition and classification** (NERC), an information extraction problem in which the goal is to recognize named en-

tities (NE) in a sentence and categorize them, for instance, in one of the following broad categories: person, location, organization or miscellaneous. The CoNLL-2003 task concerned this problem, with the above four categories, for the English language, in the context of news articles from the Reuters corpus.

5 Feature vector representation

In this section we describe the representation functions ϕ_{w} and ϕ_{p}, which respectively map a word or a phrase and their local context into a feature vector in $\mathbb{R}^n$, which in practice is a binary space. First we define a set of primitive functions which apply to words or sequences of words:

- **Word(**w**)**: The form of word w.
- **PoS(**w**)**: The part-of-speech tag of word w.
- **ChunkTag(**w**)**: The chunk tag of word w.
- **OrthoFlags(**w**)**: Binary flags of word w with regard to how is it capitalized (*initial-caps*, *all-caps*), the kind of characters that form the word (*contains-digits*, *all-digits*, *alphanumeric*, *Roman-number*), the presence of punctuation marks (*contains-dots*, *contains-hyphen*, *acronym*), single character patterns (*lonely-initial*, *punctuation-mark*, *single-char*), or the membership to a predefined class (*functional-word*, or pattern (*URL*).
- **OrthoTag(**w**)**: A tag with regard to orthographic features, which is either *capitalized* (`C`), *lowercased* (`l`), *functional* (`f`), *punctuation mark* (`.`), *quote* (`'`) or *other* (`x`).
- **Affixes(**w**)**: The prefixes and suffixes of the word w (up to 4 characters).
- P**-Gram(**$[w_s, \ldots, w_e]$**)**: The conjunction of the outputs of the primitive P on words $w_s, \ldots, w_e$. We work with Word-Grams, PoS-Grams, and OrthoTag-Grams. For instance, the OrthoTag-Gram for "John Smith payed 3 euros" is `CClxl`.

For representing *words* (function ϕ_{w}), we compute primitives in a *window* of words around x_i, i.e., words x_{i+l} with $l \in [-L_{\mathrm{w}}, +L_{\mathrm{w}}]$. Each primitive label, together with each relative position l and each returned value forms a final binary indicator feature. Specifically, we compute: (i) Word and PoS primitives; (ii) PoS-Grams on all sequences within the window which include the central word; (iii) For clausing, also ChunkTag primitives (given in the input); (iv) For NERC, also OrthoFlags, Affixes, and OrthoTag-Grams on all sequences which include the central word; (v) Left Start-Ends: flags indicating whether the words in $[-L_{\mathrm{w}}, -1]$ have been predicted as *start* and/or *end* words of a k-phrase, $k \in \mathcal{K}$.

For representing *phrases* (function ϕ_{p}), we capture the context of the phrase and the phrase itself. For the context, we evaluate: (i) A $[-L_{\mathrm{p}}, 0]$ window of primitives at the s word and a separate $[0, +L_{\mathrm{p}}]$ window at the e word. These windows include Words and PoS. On clausing, also ChunkTag features. On

NERC, also OrthoFlags and OrthoTag-Grams; (ii) Elements to the left of the s word (only for chunking and NERC). An element is either an already recognized phrase (we reduce its words into a single element and represent it by the phrase type) or a word outside a phrase (represented by its PoS). We consider up to 3 elements, and codify the relative position of each one, and also all conjunctions.

As for the (s, e) phrase itself, we evaluate primitives in $[w_s, \ldots, w_e]$ without capturing the relative position. Specifically we consider: (i) The length of the phrase; (ii) Word and PoS primitives; (iii) PoS-Grams on all subsequences of up to size 3, and also the complete PoS-Gram on the whole sequence; (iv) Concatenation of the relevant elements in the sequence (only for clausing), including among others: punctuation marks and coordinate conjunctions, the word "that", relative pronouns, verb phrase chunks, and the top clauses within the sequence, already recognized through the bottom up search; (v) Only for NERC, Word-Gram of the whole phrase, OrthoTag-Grams on subsequences of sizes 2, 3 and 4, and the Affixes of each word.

6 System implementation and experiments

The final implementation of our system makes use of the *voted perceptron* algorithm (Freund & Schapire 1999), instead of the traditional perceptron. Here, each perceptron vector generated during training is associated with a weight, which tracks the number of correct positive decisions predicted by the vector. Then, when testing, the final prediction is an *averaged* vote over the predictions of each vector. As a secondary issue, the same paper develops the dual formulation of the perceptron, which allows the use of *kernel functions*. In this work we used standard *polynomial kernels* of degree 2. Initial experiments on the problems showed poor performance for the linear case (specially on clause identification) and no significant improvements for higher degrees.

We performed experiments with our system on each problem described in Section 4. For chunking and clause identification we set a pair of *start-end* functions for each type of phrase, whereas for NERC we set only two *start-end* functions which were shared among all NE types. On each problem, we ran the FR-Perceptron for 25 epochs on the training data, and evaluated the performance on the development sets to select the optimal point in terms of F_1. Feature windows were also adjusted using the development set: for chunking we set both L_{w} and L_{p} to 2, whereas for the two other problems we set them to 3. As a general behavior, on each problem the global performance substantially increased during the first 5 epochs, and then became somewhat stable, with minor improvements (Figure 2 plots the learning curve on clause identification). Table 1 shows the obtained performance on each problem. The results are fairly good in all cases.

On chunking, we obtained a very good performance of 93.79 which is close to the two best works published on the data. Kudo & Matsumoto (2001) per-

		development			test		
	T	precision	recall	F_1	precision	recall	F_1
chunking	08	93.83%	92.77%	93.30	94.20%	93.38%	93.79
clause id.	20	90.56%	85.73%	88.08	88.17%	82.10%	85.03
NERC	12	89.59%	88.17%	88.87	83.93%	83.43%	83.68

Table 1: *Results of the three problems on the development and test sets*

type	precision	recall	F_1	type	precision	recall	F_1
ADJP	83.80%	68.49%	75.38	ADVP	85.29%	79.68%	82.39
CONJP	50.00%	44.44%	47.06	INTJ	100.00%	100.00%	100.00
LST	00.00%	00.00%	00.00	NP	94.55%	94.37%	94.46
PP	96.50%	98.13%	97.31	PRT	78.82%	63.21%	70.16
SBAR	90.81%	79.44%	84.75	VP	93.90%	93.22%	93.56

Table 2: *Detailed results on the chunking task*

formed several taggings with SVM classifiers, which were later combined. They report a performance of 93.85 with an individual tagging and 93.91 by combining many taggings. Their system makes use of several hundreds of SVM classifiers applied to each word, whereas we only need 22 perceptrons for filtering words and 11 perceptrons for scoring phrases. In contrast, their feature space is simpler than ours, since we exploit rich features on phrases. The best work on the data is (Zhang et al. 2002), which applied regularized Winnow. They report a base performance of 93.57, and an improved result of 94.17 by using external grammatical information. Table 2 shows the performance of our chunker on each individual phrase type. Looking at the recognition of Noun Phrases (NP), our system, with $F_1 = 94.46$, slightly outperforms recent systems trained specifically for this chunk. Kudo & Matsumoto (2001) obtained 94.39 with combination of SVMs. Sha & Pereira (2003) obtained 94.38 with conditional random fields. They also report 94.09 for the perceptron-based tagger by Collins (2002).

On clause identification, our system improves the performance of the best system published so-far (Carreras et al. 2002), which obtained $p = 90.18\%$, $r = 78.11\%$, and $F_1 = 83.71$ on the test set. That system made use of the same phrase recognition model, and the decisions were learned separately by AdaBoost classifiers. Additionally, the scoring function was a robust combination of several classifiers.

On NERC, we obtained a performance of 83.68 on the test set, which is competitive, given the basic feature we used, but still far from the top systems of the competition, which achieved above 88 in F_1. It seems that the feature engineering and the use of external resources, such as large gazetteers, allows to substantially improve the results on this problem.

To get a clearer picture of the FR-Perceptron learning strategy, we were interested in comparing it against usual alternatives, for clause identification. We first

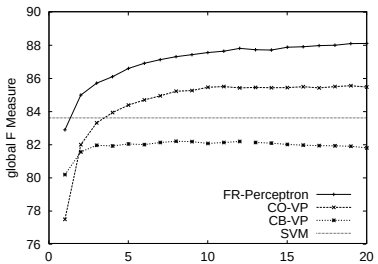

Figure 2: *Evolution of the performance on clause identification*

considered a batch classification setting, in which each function is trained separately via binary classification loss. To do so, we generated three datasets from training sentences, one for each function. For the *start-end* sets, each word in the data formed a training example. For the *score* set, we generated only phrases formed by correct *start-end* pairs. In this setting, we trained the functions with the voted perceptron batch algorithm for binary classification (CB-VP), and also as Support Vector Machines (SVM). Secondly, we considered an online alternative in which the functions are trained together via binary classification loss. As in the FR-Perceptron the score function is trained with respect to the actual behavior of the start-end functions, but here binary classification feedback is given instead of recognition-based feedback. We refer to this model as CO-VP.

Figure 2 shows the performance curve on the development set in terms of the F_1 measure with respect to the number of epochs during training. Clearly, the behavior of the FR-Perceptron is much better than the others, being at any epoch 2 points better than the online CO-VP model, and far from the batch models. It seems quite evident that online models perform better than batch models because they are able to capture the interaction between the filtering and ranking layers. Furthermore, looking at the difference between FR-Perceptron and CO-VP, this interaction is better exploited with the proposed recognition-based feedback than with the usual binary classification feedback.

7 Conclusions

We have presented a learning architecture for general phrase structure recognition. The method makes use of several decision functions operating in two layers: at word level, to identify phrase candidates, and at phrase level, to score the optimal ones. Doing so, we are able to incorporate rich features which represent partial structures of the solution. The main contribution of the work is to propose a simple online learning algorithm for training, at the same time, all the involved functions in the form of voted perceptrons.

We have empirically proved the generality and feasibility of the approach by

applying it to different phrase recognition problems on named entity recognition and partial parsing, in which we achieve the state-of-the-art performance. The experimentation evinces that exploiting the interaction between learned functions during learning results in a better global behavior.

Acknowledgements. This research has been supported by the European Commission (Meaning, IST-2001-34460) and the Spanish Research Department (Hermes, TIC2000-0335-C03-02; Petra, TIC2000-1735-C02-02). Xavier Carreras holds a grant from the Catalan Research Department.

REFERENCES

Carreras, Xavier, Lluís Màrquez, V. Punyakanok & Dan Roth. 2002. "Learning and Inference for Clause Identification". *Proceedings of the 14th European Conference on Machine Learning* (*ECML'02*), 35-47. Helsinki, Finland.

Carreras, Xavier & Lluís Màrquez. 2003. "Phrase Recognition by Filtering and Ranking with Perceptrons". *Proceedings of the Intl. Conference on Recent Advances in Natural Language Processing* (*RANLP'03*), 78-85. Borovets, Bulgaria.

Collins, Michael. 2002. "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms". *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Procesing* (*EMNLP*), 1-8. Philadelphia, U.S.A.

Crammer, Koby & Yoram Singer. 2003. "A Family of Additive Online Algorithms for Category Ranking". *Journal of Machine Learning Research* 3:1025-1058.

Freund, Yoav & Robert E. Schapire. 1999. "Large Margin Classification Using the Perceptron Algorithm". *Machine Learning* 37:3.277-296.

Har-Peled, Sariel, Dan Roth & Dav Zimak. 2003. "Constraint Classification for Multiclass Classification and Ranking". *15th Conference on Neural Information Processing Systems* (*NIPS-15*), 785-792. Vancouver, Canada. MIT Press.

Kudo, Taku & Yuji Matsumoto. 2001. "Chunking with Support Vector Machines". *Proceedings of the 2nd Conference of the North American Chapter of the Association for Computational Linguistics* (*NAACL'01*), 192-199. Pittsburgh, Pennsylvania.

Punyakanok, Vasin & Dan Roth. 2001. "The Use of Classifiers in Sequential Inference". *Proceedings of the 13th Conference on Neural Information Processing Systems* (*NIPS-13*), 995-1001. Denver, Colorado.

Ratnaparkhi, Adwait. 1999. "Learning to Parse Natural Language with Maximum-Entropy Models". *Machine Learning* 34:1.151-175.

Sha, Fei & Fernando Pereira. 2003. "Shallow Parsing with Conditional Random Fields". *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference* (*HLT-NAACL'03*), 213-220. Edmonton, Canada.

Zhang, Tong, Fred Damereau & David E. Johnson. 2002. "Text Chunking Based on a Generalization of Winnow". *Journal of Machine Learning Research* 2:615-637.

Cascaded Finite-State Partial Parsing: A Larger-First Approach

SEBASTIAN VAN DELDEN* & FERNANDO GOMEZ**

*University of South Carolina Upstate
**University of Central Florida

Abstract

A larger-first approach to partial parsing is presented. This approach is opposite to current easy-first approaches, and is capable of producing a more detailed partial parse. Syntactic relations are identified in descending order of their size. Special consideration is given to the comma punctuation mark. Its presence is used in identifying large syntactic relations and establishing a boundary for containment of ambiguity. The system is tested on the Wall Street Journal section of the Penn Treebank III and compared against a well-known easy-first approach.

1 Introduction

A growing trend in natural language processing is to decompose a parser into intermediate components. First, part-of-speech information is assigned to each word in the sentence. Next, a partial parse of the sentence identifying larger phrases and clauses is produced. Finally, a semantic analysis is performed to determine verb meaning and thematic roles, and resolve attachment issues. Unlike part-of-speech tagging (Brants 2000; Brill 1994) and semantic interpretation (Gomez 2001), no formal standard has been defined which establishes the detail or *richness* that a partial parser can and should achieve. We present a larger-first approach to partial parsing which is opposite to current easy-first approaches. Syntactic relations are identified primarily in a descending order of their size. The approach is implemented with specialized sets of deterministic finite state automata that assign a hierarchy of tags to the tokens in the sentence.

Abney (1996) defines an easy-first finite state approach to partial parsing (named CASS[1]) in which smaller syntactic relations, like noun and verb phrases, are identified first then combined to form larger syntactic relations, like prepositional phrases and relative and subordinate clauses. The larger-first approach uses simple deterministic finite state automata to first identify larger relations and then focus on the smaller relations comprising them.

The first step in larger-first partial parsing is to determine the syntactic roles of commas (van Delden & Gomez 2004, 2003; Bayraktar et al. 1998), since commas are usually used to delimit or comprise large syntactic relations. Jones

[1] CASS Version 1h was used in our experiments

(1994) notes that the comma is the most abundant punctuation mark in Wall Street Journal Penn Treebank (Marcus et al. 1993). A closer analysis of the Wall Street Journal Section of the Penn Treebank III reveals that 65% of all sentences contain at least one comma with: 20% containing two commas; 9% containing three commas; 4% containing four commas, and almost 2% containing five or more commas. We have found that the syntactic roles of commas can be determined with a 95% accuracy when tested on this corpus. A specialized network of automata for commas is therefore a good foundation on which the rest of the partial parse should be built. Consider the following sentence: *Some seals, such as the leopard seal(Hydrurga leptonyx), are quite predatory, feeding on penguins, other birds that land on water, and other seals.* A partial parsing system must pay special attention to the roles that are being played by the commas in this sentence in order to realize that there is a relative clause (introduced by a comma) which contains a list of noun phrases that in turn contains another embedded relative clause. Such sentences are abundant in the Penn Treebank III.

In this approach: 1) explicit attachment decisions are always avoided, and 2) containment of attachment ambiguity may only occur within comma-delimited syntactic relations. Containment of ambiguity refers to limiting the attachment sites of a syntactic relation to its consuming clause. For example, *I picked up the hammer, (REL which lay next to the nails on the table).* The attachment sites of the prepositional phrases *next to the nails* and *on the table* are limited to within the comma-delimited relative clause. Also consider the sentences: *Many foreigners came to Hawaii to work on the plantations* versus *Many foreigners came to Hawaii to work via transportation vessels.* It cannot be determined based on syntax alone where the final prepositional phrase should be attached. Therefore *on the plantations* and *via transportation vessels* are left unattached - they could be attached within the infinitival clause or to another peer syntactic relation in the sentence, like the main verb *came*.

Besides the ability to accurately disambiguate commas, the larger-first approach is also preferred because it is capable of producing a richer partial parse than an easy-first approach. *Richness* of a partial parse refers to the level of detail a partial parsing system produces. The larger-first approach is capable of identifying appositions and partially disambiguating coordinate conjunctions. An easy-first approach, however, cannot be extended to provide such richness (see Section 2 for more details).

The remainder of the paper is organized as follows: Section 2 motivates the need for the larger-first approach; Section 3 describes the syntactic relation set; Section 4 presents the larger-first partial parsing algorithm; Section 5 evaluates the performance of the system and compares these results to the CASS system; and Section 6 concludes the paper.

2 Motivation

An easy-first approach is unable to provide the detail of a larger-first partial parser. In particular, attempting to identify appositions or partially resolve coordination ambiguity will violate the easy-first approach. The larger-first approach is also preferred due to the ambiguity of the part-of-speech tags (Santorini 1995) used by most parsing systems.

Attempting to disambiguate appositionsfrom lists of noun phrases would violate the easy-first approach. Syntactically, an apposition is (usually) a noun phrase enclosed by commas that appositives a preceding noun phrase. This relation is syntactically smaller than a list of noun phrases which is composed of at least three noun phrases, one or two commas, and a conjunction. Consider the following sentence: *John eats a banana, an apple, or a pear for breakfast. An apple* would be incorrectly identified as an apposition by the easy-first approach since syntactically smaller relations are identified prior to larger relations. This would occur whenever there is a list of at least three noun phrases with a comma before the conjunction. The easy-first approach cannot be extended to make this distinction since smaller relations are always identified prior to larger relations.

Unlike the larger-first approach, the easy-first approach is also incapable of being extended to partially disambiguate coordinate conjunctions. Here we define partial disambiguation as identifying the post-conjunct of a coordinate conjunction. For example, *We sold the car with the cloth interior and the truck after our boss left.* The conjunction is identified as coordinating a noun phrase. However, no attempt is made to identify the pre-conjunct (*the car* or *the cloth interior*). Partial disambiguation is reasonable at this point since a semantic algorithm (Argawal & Bogess 1992) can be used to determine the pre-conjunct afterwards.

Baker (1995) suggests that coordination can be resolved by identifying the largest relation of similar syntax on either side of the conjunction. The larger-first approach is ideal for accomplishing such a task. Consider the following sentence: *The boys went to the beach and the girls went to the mall*. The larger-first approach identifies the larger syntactic relations on either side of the conjunction first and is capable of determining that the conjunction coordinates two independent clauses. An easy-first approach, however, would incorrectly identify the conjunction as coordinating the smaller syntactic relations surrounding the conjunction - the noun phrases.

The larger-first approach is also better at handling ambiguity that can be present in part-of-speech tags. For example, the Penn Treebank Tagset (Santorini 1995) provides one tag (*IN*) to identify either prepositions or subordinate conjunctions. Following an easy-first approach, will result in an error whenever a subordinate conjunction which could also be a preposition is present. For example, *John went to the black board after the teacher threatened to expel him.*

In the easy-first approach, *after the teacher* would first be identified as a prepositional phrase. This error could possibly be changed later during processing. The larger-first approach identifies the syntactically larger subordinate clause *after the teacher threatened to expel him first*, no later changes are needed.

3 The Syntactic Relation Set

The larger-first approach is broken down into four specialized networks of deterministic finite state automata: 1) the Comma Network; 2) the Conjunction Network; 3) the Clause Network; and 4) the Phrase Network.

The comma network identifies syntactic relations that are associated with the commas in the sentence. Commas are usually used to delimit a syntactic relation. In this case, the boundary of the syntactic relation is marked by a comma, the start of the sentence, or the end of the sentence. However, when commas are used to coordinate items in a series, a boundary needs to be established. This boundary is established by avoiding any type of attachment decisions. For example, *Thomas purchased the house in* ***the country, the yacht with the 20 foot sail, and the sports car*** *with the sunroof*. Only the highlighted words are included in the list of noun phrases. *With a sun roof* is not included in the list of noun phrases, because syntactically we cannot determine where it should be attached. In this case it should be attached to *the sports car*, but if it is replaced by *with the money*, then the attachment should occur at the verb. Our definition of partial parsing is not violated - *with the sunroof* can be attached to a sub clause (the list of noun phrases) or a peer (the verb). Note, however, that some attachment ambiguity has been contained. Attachment sites for *with the 20 foot sail* are limited to within the marked list of noun phrases.

The conjunction network identifies syntactic relations that are being coordinated by a coordinate conjunction with no preceding comma. Unlike the comma network, the boundaries for the conjunction network are very often influenced by attachment decisions. Conjunctions are only partially disambiguated by identifying the type of syntactic relations that are being coordinated, leaving the location of the pre-conjunct unresolved. For example, *We sold the car with the cloth interior* ***and the truck*** *in the showcase*. *And the truck* is identified as a noun phrase that is being coordinated. However, the pre-conjunct (*the car*) is not recognized and the prepositional phrase (*in the showcase*) is not attached even though it appears directly to the right of *the truck*. This attachment requires semantic information (Agarwal & Boggess 1992) and should not be performed by a primarily syntactic partial parser. For example, the sentence could have been *We sold the car with the cloth interior and the sunroof in the showcase*. Even though these sentences are syntactically equivalent, the pre-conjunct is now *the cloth interior* and *in the showcase* should be attached *to the car*.

Boundary identification by the clause network is also directly influenced by

attachment issues. For example, consider *that*-clauses. No attempt is made to classify a that-clause as a relative or subordinate clause even though the distinction is made in the part-of-speech tag that is assigned to it: WDT (relative determiner) or IN (preposition or subordinate conjunction). Syntactic part-of-speech taggers, however, are not capable of correctly making this distinction. For example, syntactically it cannot be determined that *The news that Mary bought the car was exciting* contains a subordinate clause and *The news that Mary brought the family was exciting* contains a relative clause. Furthermore, the exact boundary of the that-clause cannot be determined without semantics. In our approach, the boundary of a that-clause is reached when an attachment issue is encountered. For example, *I gave the Tulips that I bought in Holland to the children. In Holland* and *to the children* are not included within the that-clause. A semantic analysis is needed to determine that *in Holland* should be included and *to the children* should not be included.

4 The larger-first algorithm

If an automaton at any level in the cascade accepts, it assigns *a structural tag*to the words that were consumed by specifically marked arcs. The $>$ symbol is used as a prefix to indicate that the current token is grouped with the token that follows it. No $>$ prefix marks the end of the syntactic relation. This notation has been used in several partial parsing systems (Ramshaw & Marcus 1995; Voutilainen & Jarvinen 1995) to introduce a new, single layer of tags. However, it is extended here to the next logical step - representing a partial tree structure of multiple levels within the structural-tags. The larger-first partial parsing algorithm is shown in Figure 1.

The speed of the algorithm is dependant on the size of the sentence, number of automata, and number of levels created during the parse. However, since the number of automata is fixed and the number of levels of structural tags is relatively small (4 or 5), speed depends primarily of the size of the input sentence alone.

Each automaton is processed on every level of tags. Initially, there is only one level of tags - part-of-speech tags assigned to every token by a tagger. Each specialized network is capable of introducing one or more new levels of structural-tags to the sentence. In the larger-first approach, the networks are considered in descending order of the size of the syntactic relations that they identify. First, the comma network is considered, then the conjunction network, followed by the clause network, and finally the phrase network.

Within a specialized network, conflicting automata are also considered in descending order of size. Consider the following sentence that has been part-of-speech tagged: *Peter/NNP eats/VBZ a/DT banana/NN ,/, an/DT apple/NN ,/, or/CC an/DT orange/NN for/IN breakfast /NN ./.* After part-of-speech tagging,

foreach network$_i$
  foreach automaton$_{ij}$ that assigns structural tag$_j$
    foreach level$_k$
      try automaton$_{ij}$ at each position n on level$_k$
      if automaton$_{ij}$ accepts at $n + m$
        insert structural tag$_j$ at level$_k$ to the designated input tokens and continue processing at the position $n + m + 1$ of level$_k$
      else
        continue processing at position $n + 1$ of level$_k$

Figure 1: *The larger-first partial parsing algorithm*

this sentence has one level of tags and would initially be passed to the comma network. The sentence contains two possible syntactic relations that could be identified by the comma network: a list of noun phrases or an apposition. A conflict therefore arises between the automata that recognize these different syntactic relations. The conflict is resolved by identifying the syntactic relations in descending order of their size. The list of noun phrases would therefore be recognized prior to attempting to recognize appositions. A new level of structural tags would be assigned by the list of noun phrases automaton: *Peter/NNP eats/VBZ a/>LST-NP/DT banana/>LST-NP/NN ,/>LST-NP/, an/>LST-NP/DT apple/>LST-NP/NN ,/>LST-NP/, or/>LST-NP/CC an/>LST-NP/ DT orange/LST-NP/NN for/IN breakfast/NN ./.* The first level of tags now includes structural as well as part-of-speech tags, and the apposition automata will not be able to (incorrectly) identify an apposition on the first level.

Automata across or within a network are capable of utilizing structural tags that have been assigned by previous automata, relaxing the larger-first rule. Consider the following example that has been part-of-speech tagged: *President/NNP Bush/NNP will/MD ,/, after/IN much/JJ debate/NN ,/, postpone/VB a/DT war/NN with/IN Iraq/NNP ./.* The sentence has one level of tags and is passed to the comma network which identifies a prepositional phrase that is enclosed by commas: *President/NNP Bush/NNP will/MD ,/>CO-PP/, after/>CO-PP/IN much/ >CO-PP/JJ debate/>CO-PP/NN ,/CO-PP/, postpone/VB a/DT war/NN with/IN Iraq /NNP ./.* Note, however, that this syntactic relation splits the verb phrase *will postpone*. The verb phrase can be recognized by simply including the *CO-PP* structural tag in the verb phrase automaton. When this sentence is passed to the phrase network, the first level of tags will be: *NNP MD >CO-PP >CO-PP >CO-PP >CO-PP CO-PP VB DT NN IN NNP*. On this first level, the verb phrase *will, after much debate, postpone* is easily recognized by including a self arc that can be taken on the *CO-PP* tag in between the modal and the main verb: *MD CO-PP* VB*. This minor adjustment allows the verb phrase automaton to identify the verb phrase. A new

level of appropriate structural tags(*CO-VP*) would be introduced: *President/NNP Bush/NNP will/>CO-VP/MD ,/>CO-VP />CO-PP /, after/ >CO-VP />CO-PP /IN much/>CO-VP />CO-PP /JJ debate/ >CO-VP />CO-PP /NN,/>CO-VP /CO-PP/, postpone/ CO-VP/VB a/DT war/NN with/IN Iraq/NNP ./.*

Consider the following example sentence: *We ought to be phasing out the estate tax, and we should work toward a tax code that is unique and modern.* The comma network first recognizes that an independent clause (CO-S) is present in the sentence, and introduces a new level of appropriate structural-tags. The remaining networks are now processed on both levels one and two. The conjunction network recognizes that an adjective is being coordinated (CC-JJ) on the second level and introduces a new layer of appropriate structural-tags. The clause network now has three levels to process. An infinitival clause is recognized on the first level and a that-clause on the second level. Note that the that-clause automaton consumes the tags that were assigned by the conjunction network (CC-JJ). This is an example of where a smaller syntactic relation is identified first and then combined to form a larger one. Finally the phrase network processes all four levels and groups together any remaining syntactic relations. The final output is:

```
LEVELS:   1       2      3       4       5

We         NP      PRP
ought      VP      MD
to        >INF     TO
be        >INF    >VP     VB
phasing   >INF    >VP     VBG
out       >INF     VP     RP
the       >INF    >NP     DT
estate    >INF    >NP     NN
tax        INF     NP     NN
,          ,
and       >CO-S    CC
we        >CO-S    NP     PRP
should    >CO-S   >VP     MD
work      >CO-S    VP     VB
toward    >CO-S   >PP     IN
a         >CO-S   >PP    >NP      DT
tax       >CO-S   >PP    >NP      NN
code      >CO-S    PP     NP      NN
that      >CO-S   >THT    WDT
is        >CO-S   >THT    VP      VBZ
unique    >CO-S   >THT   >PRED    JJ
and       >CO-S   >THT   >PRED   >CC-JJ   CC
modern     CO-S    THT    PRED    CC-JJ   JJ
.          .
```

5 Evaluation

The larger-first approach (hereafter referred to as LAFI — LArger-FIrst) was evaluated on Section 23 of the Wall Street Journal Penn Treebank III (Marcus et al. 1993). The CASS system was also evaluated on this corpus. The results are shown in Table 1.

System	Accuracy	Richer Partial Parse	Better Ambiguity Containment
CASS	87.5%	–	5.1%
LAFI	88.6%	31.3%	36.3%

Table 1: *Evaluation and comparison of the CASS and LAFI systems*

CASS was compared against LAFI by following three evaluation criteria: 1) accuracy on the sentence level, 2) richness or detail of partial parse, and 3) containment of ambiguity.

5.1 *Sentence level accuracy*

CASS and LAFI identify many similar syntactic relations, motivating a comparison between the two approaches. However, LAFI identifies appositions and partially disambiguates conjunctions which CASS is incapable of identifying. Furthermore, the exact boundaries of the syntactic relations and structure of partial parse differs across systems. Sentence level accuracy is used here to generate a rough comparison of both systems based on their own criteria and syntactic relation set. Since the CASS system identifies a different set of syntactic relations than LAFI, it would be illogical to compare them on the syntactic relation level. Instead, an error was assigned if either system made a mistake on the sentence level. As shown in Table 1, the evaluation yielded a similar accuracy for both systems: 87.5% for CASS and a slightly better 88.6% for LAFI. This measure is intended here only as an indicator that LAFI is capable of producing an accuracy comparable to the CASS system.

The main contribution of the larger-first approach is that it is capable of producing a richer partial parse and a better containment of ambiguity while maintaining an accuracy comparable to that of the easy-first approach.

5.2 *Partial parse richness*

The richness of a partial parse refers to the level of detail a partial parsing system produces. A system that disambiguates more syntactic relations than another system delivers a richer partial parse. Sentences that were correctly partially parsed by both systems' standards were analyzed to see which system created a more detailed partial parse. As shown in Table 1, LAFI produces a richer partial

parse on 31.3% of the test sentences, indicating that LAFI is capable of achieving a more detailed partial parse than CASS while remaining as accurate as the CASS system.

5.3 *Containment of ambiguity*

Sentences that were correctly partially parsed by both CASS and LAFI were also analyzed for the extent of containment of ambiguity. Since LAFI uses comma information to help limit attachment sites, containment of attachment ambiguity is often much better in sentences containing commas. As shown in Table 1, the LAFI system contained ambiguity of attachment sites better than the CASS system for 36.3% of the test sentences - primarily because of the information provided by the comma punctuation mark. In some situations, CASS performs a right-most attachment of prepositional phrases. In these cases, the CASS system produces better containment of attachment ambiguity than LAFI (if the attachment is correct) since LAFI does not resolve any explicit attachment decisions. If an attachment is incorrect an error is generated (Section 5.1). The 5.1% better containment of ambiguity for CASS corresponds to the simple right-most attachment of prepositional phrases that is sometimes performed by the CASS system.

6 Conclusions

A larger-first approach to partial parsing which is opposite to current easy-first approaches has been presented. Syntactic relations are identified primarily in descending order of their size. The accuracy of the larger-first approach is comparable to that of the easy-first approach while: 1) producing a more detailed partial parse than the easy-first approach is incapable of producing; and 2) providing a better containment of ambiguity by identifying the syntactic roles of commas early in the parse. An algorithm has been presented that incrementally applies each set of specialized automata to an input sentence, capturing a partial tree structure with multiple levels of structural tags that are assigned to each word in the sentence.

Acknowledgements. We thank the National Aeronautics and Space Administration for their funding for this research - NASA grant 16-40-202.

REFERENCES

Abney, Steven. 1996. “Partial Parsing via Finite State Cascades”. *Proceedings of the Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information*, 8-15. Prague, Czech Republic.

Agarwal, Rajeev & Lois Boggess. 1992. "A Simple But Useful Approach to Conjunct Identification". *Proceedings of the 30th Meeting of the Association of Computational Linguistics* (*ACL'92*), 15-21. Univ. of Delaware, Newark.

Baker, Carl. 1995. *English Syntax, Second Edition*. Cambridge, Mass.: MIT Press.

Bayraktar, Murat, Bilge Say & Varol Akman. 1998. "An Analysis of English Punctuation: The Special Case of Comma". *International Journal of Corpus Linguistics* 3:1.33-57.

Brants, Thorsten. 2000. "TnT: A Statistical Part-of-Speech Tagger". *Proceedings of the 6th Applied Natural Language Processing Conference* (*ANLP-2000*), 224-231. Seattle, Washington.

Brill, Eric. 1994. "Recent Advances in Transformation Based Error-Driven Learning". *Proceedings of the ARPA Human Language Technology Workshop*, 722-727. Princeton, N.J.

Gomez, Fernando. 2001. "An Algorithm for Aspects of Semantic Interpretation Using an Enhanced WordNet". *Proceedings of the 2nd Meeting of the North American Chapter of the Association of Computational Linguistics*, 87-94. Carnegie Mellon Univ., Pittsburgh.

Jones, Bernard. 1994. "Towards Testing the Syntax of Punctuation". *Proceedings of the 34th Meeting of the Asoociation of Computational Linguistics* (*ACL'94*), 363-365. Santa Cruz, California.

Marcus, Mitchell, Beatrice Santorini & Mary A. Marcinkiewicz. 1993. "Building a Large Annotated Corpus of English: The Penn Treebank". *Computational Linguistics* 19:2.313-330.

Ramshaw, Lance & Mitch Marcus. 1995. "Text Chunking using Transformation-Based Learning". *Proceedings of the 3rd Workshop on Very Large Corpora* ed. by D. Yarovsky & K. Church, 82-94. Somerset, New Jersey.

Santorini, Beatrice. 1995. *Part-of-Speech Tagging Guidelines for the Penn Treebank Project*. 3rd Revision, 2nd Printing. Technical report, Dept. of Computer & Information Science, Univ. of Pennsylvania.

van Delden, Sebastian & Fernando Gomez. 2004. "A Finite-State Comma Tagger". *The International Journal of Artificial Intelligence Tools* 13:3. World Scientific Publishing Company.

van Delden, Sebastian & Fernando Gomez. 2003. "Extending a Finite State Approach for Parsing Commas in English to Dutch". *Proceedings of the 13th Meeting of Computational Linguistics in the Netherlands. Selected Papers from the CLIN Meeting*. (= *Language and Computers: Studies in Practical Linguistics*, 47), 25-38. Amsterdam, The Netherlands: Rodopi Publishers.

Voutilainen, Altro & Timo Jarvinen. 1995. "Specifying a Shallow Grammatical Representation for Parsing Purposes". *Proceedings of the 7th Meeting of the European Chapter of the Association of Computational Linguistics*, (*EACL'95*), 210-214. Dublin.

A Constraint-based Bottom-up Counterpart to Definite Clause Grammars

HENNING CHRISTIANSEN

Roskilde University

Abstract

A new grammar formalism, CHR Grammars (CHRG), is proposed that provides a constraint-solving approach to language analysis, built on top of the programming language of Constraint Handling Rules in the same way as Definite Clause Grammars (DCG) on Prolog. CHRG works bottom-up and adds the following features when compared with DCG: (i) An inherent treatment of ambiguity without backtracking. (ii) Robust parsing; do not give up in case of errors but return the recognized phrases. (iii) A flexibility to produce and consume arbitrary hypotheses making it straightforward to deal with abduction, integrity constraints, operators *à la* assumption grammars, and to incorporate other constraint solvers. (iv) Context-sensitive rules that apply for disambiguation, coordination in natural language and tagger-like rules.

1 Introduction

Definite Clause Grammars (Colmerauer 1975, Pereira & Warren 1980) (DCG) have been appreciated for their declarative nature and execution environment inherited from the logic programming language Prolog. Where Prolog and DCG work top-down, the language of Constraint Handling Rules (Frühwirth 1998) (CHR) provides a logic programming framework for bottom-up computations that implies several advantages for language processing (Abdennadher & Schütz 1998, Abdennadher & Christiansen 2000). In fact, any context-free grammar or DCG can be rewritten in a straightforward way as a set of propagation rules of CHR that serves as an error robustparser with an inherent treatment of ambiguitywithout backtracking, and no artificial nonterminals are necessary as often in a DCG. Restrictions are that empty productions and loops among nonterminals cannot be handled.

A new and powerful grammar formalism, called CHRG for CHR Grammars, is proposed with a form of context-sensitive rules that can take into account arbitrary grammar symbols to the left and right of a sequence supposed to match a given nonterminal. This allows tagger-like grammar rules, it can be used for disambiguating simple and ambiguous context-free grammar rules, and provides also a way to handle coordination in natural language as shown by an example: The following rules are excerpt of a CHR grammar for sentences such as "*Peter likes and Mary detests spinach*".

```
sub(A), verb(V), obj(B) ::> sent(s(A,V,B)).
subj(A), verb(V) /- [and], sent(s(_,_,B))
                                    ::> sent(s(A,V,B)).
```

The first rule is to be understood in the usual way that a complete `sub-verb-obj` sequence can be reduced to a `sent` node. The second rule is an example of a context-sensitive rule: It applies to a `subj-verb` sequence only when followed by terminal symbol "`and`" and another `sent` node, and in this case the incomplete sentence takes its object, matched by variable `B`, from this following `sent` node. The marker "`/-`" separates the `subj-verb` sequence from the required right context; a similar marker may indicates left context. In contrast to most other notations, CHRG mentions the constituents before the whole to emphasize the bottom-up nature.

The CHRG notation includes the full expressive power of CHR, including the ability to integrate with arbitrary constraint solvers and a highly flexible way to handle sets of extra-grammatical hypotheses. For example, abduction for context comprehension can be characterized in CHRG in a surprisingly simple way that requires no meta-level overhead as do other approaches to abduction in language processing. Elements of linear logic as in Assumption Grammars (Dahl et al. 1997) are included in a similar way; for reasons of space, these and other facilities are not presented here but we refer to the web site for CHRG (Christiansen 2002b) with comprehensive set of examples and to the larger journal paper (Christiansen 2005).

2 Background and related work

The notion of constraints, with slightly different meanings, is often associated with language processing. "Constraint grammars" and "unification grammars" are often used for feature-structure grammars, and constraint programming techniques have been applied for the complex constraints that arise in natural language processing; see, e.g., (Allen 1995, Duchier 2000) for introduction and overview. See also (Blache 2000, Duchier & Thater 1999, Maruyama 1990, Schröder et al. 2000) for similar approaches.

CHR has been applied for diagram parsing by Meyer (2000) but not elaborated into a grammar formalism; Morawietz (2000) has implemented deductive parsing (Shieber et al. 1995) in CHR and shown that a specialization of a general bottom-up parser leads to rules similar to those produced by our translator; none of these consider context in the sense we do. Abduction in CHR has been applied by (Christiansen & Dahl 2002) for diagnosis and correction of grammatical errors. An attempt to characterize the grammar of ancient Egyptian hieroglyph inscriptions by means of context-sensitive rules in CHRG is given by (Hecksher et al. 2002). CHR is available as extension of, among others, SICStus Prolog

(Swedish Institute of Computer Science 2002) which is the version applied in the present work.

3 Syntax, semantics and implementation of CHRG

A *CHR Grammar*, or *CHRG* for short consists of finite sets of *grammar* and *constraints symbols* and a finite set of *grammar rules*. An *attributed grammar symbol*, for short called a *grammar symbol*, is formed as a logical atom whose predicate symbol is a grammar symbol; a grammar symbol formed by `token`/1 is called a *terminal*, any other grammar symbol a *nonterminal*. Sequences of terminal symbols `token(`a_1`)`, ..., `token(`a_n`)` may also be written `[`a_1`, ..., `a_n`]`; if ground, such a sequence is called a *string*.

A *propagation (grammar) rule* is of the form α `-\` β `/-` γ `::>` G `|` δ. The part of the rule preceding the arrow `::>` is called the *head*, G the *guard*, and δ the body; $\alpha, \beta, \gamma, \delta$ are sequences of grammar symbols and constraints so that β contains at least one grammar symbol, and δ contains exactly one grammar symbol which is a nonterminal (and perhaps constraints); α (γ) is called *left (right) context* and β the *core* of the head; G is a guard as in CHR that may test properties for the variables in the head of the rule. If left or right context is empty, the corresponding marker is left out and if G is empty (interpreted as `true`), the vertical bar is left out. The convention from DCG is adopted that non-grammatical constraints in head and body of a rule are enclosed in curly brackets.

The implemented system combines CHRG with rules of CHR and Prolog which is convenient for defining behaviour of non-grammatical constraints. CHRG includes also notation for gaps and parallel match not described here.

In Chomskian grammars, derivations are defined over strings of symbols and this suffices also for a large class of CHRGs. Several aspects of CHRG make this too restrictive: Context-sensitive rules of CHRG extend those of Chomsky by the possibility to refer to any grammar symbol which has been created at some stage during derivation (not only the "current" stage); extra-grammatical hypotheses created during a derivation serve as a common resource for all subsequent derivation steps; CHRG includes other sorts of rules (below) inherited from CHR which need to be specified in a bottom-up fashion.

The most obvious way to define derivations in CHRG seems to be to represent sequencing by means of word boundaries (e.g., integer numbers) and each stage in the derivation as a constraint store. For each grammar symbol N of arity n, we assume a corresponding constraint also denoted by N of arity $n+2$ called an *indexed grammar symbol* with the two extra arguments referred to as phrase (or word) *boundaries*.

For a grammar symbol $S = N(\bar{a})$, the notation S^{n_0,n_1} refers to the indexed grammar symbol $N(n_0, n_1, \bar{a})$ with integers $n_0 < n_1$; in case of a ter-

minal, $n_0 + 1 = n_1$ is assumed. For any sequence σ of grammar symbols $S_1, \ldots, S_k$ and increasing integers $n_0, n_1, \ldots, n_k$, we let σ^{n_0,n_k} refer to the set $\{S_1^{n_0,n_1}, \ldots, S_k^{n_{k-1},n_k}\}$ with the existence of $n_1, \ldots, n_{k-1}$ understood. This extends so that for a sequence of grammar symbols and extra-grammatical constraints, we remove all constraints from the sequence, put indexes on the remaining grammar symbols, and add again the constraints in their original position.

A *constraint store* is a set of constraints and indexed grammar symbols and the *initial store* for a terminal string σ is the store $\sigma^{0,k}$ where k is the length of σ. An *instance* (and *ground instance*) of a grammar rule is defined in the usual way. A *derivation step* from one constraint store $\mathbf{S}_1$ to another $\mathbf{S}_2$ by an instance of a propagation grammar rule α `-\` β `/-` γ `::>` G `|` δ is defined whenever

- $\vdash \exists \bar{x} G$ where $\bar{x}$ are the variables in G not in α, β, γ,
- $\alpha^{i,j} \cup \beta^{j,k} \cup \gamma^{k,\ell} \subseteq \mathbf{S}_1$, and $\mathbf{S}_2 = \mathbf{S}_1 \cup \delta^{j,k}$.

Useful for optimization purposes, CHRG includes two other sorts of rules that reflect the underlying CHR system. A *simplification (grammar) rule* is similar to a propagation rule except that the arrow is replaced by `<:>`; a *simpagation (grammar) rule* is similar to a simplification except that one or more grammar symbols or constraints in the core of the head are prefixed by an exclamation mark "`!`". Derivation with these rules is defined as above, except that the new state is given $\mathbf{S}_2 = \mathbf{S}_1 \cup \delta^{j,k} \setminus \beta^{j,k} \cup \beta'^{j,k}$ where $\beta'^{j,k}$ are those elements of $\beta^{j,k}$ prefixed by exclamation marks.

A *parsing derivation* for terminal string σ (and given grammar) is defined as a sequence of steps starting with state $\sigma^{0,n}$. A *final* constraint store is one in which no further step can apply; for any grammar symbol N with $N^{0,n}$ in the final store, we say that N is *accepted* from σ.

CHRG is implemented by a straight-forward compiler that translates a grammar into a CHR program which, when executed, realizes the semantics defined above in the shape of a running parser and analyzer. As shown in (Christiansen 2005), execution speed depends highly on the grammar: Selected grammars run in linear time and arbitrary context-free grammar without attributes runs in cubic time similarly to classical algorithms such as Early and Cocke-Younger-Kasami.

4 Examples

The following shows the full syntax used in the implemented system. The "`handler`" command is a reminiscent of the CHR system; grammar symbols are declared by the `grammar_symbols` construct as shown. The final command has no effect in the present example, but it adds extra rules needed for the extensions described below.

```
handler my_grammar.
grammar_symbols np/0, verb/0, sentence/0.
np, verb, np ::> sentence.
```

```
[peter] ::> np.
[mary] ::> np.
[likes] ::> verb.
end_of_CHRG_source.
```

For the string "*Peter likes Mary*", the `np`, `verb`, and `np` are recognized, and then the `sentence` rule applies. This grammar consists of propagation rules, so the `token`s, `np`s, and `verb` are not consumed. If a rule were added, say `np, [likes] ::> sentence1`, a `sentence` as well as a `sentence1` would be recognized. If all rules were changed into simplification rules (changing `::>` to `<:>`), only one would be recognized. For an ambiguous grammar of propagation rules, a sentence node is generated for each different reading.

Context-sensitive rules were shown in the introduction for coordination handling. In the following, left and right contexts are applied in a tagger-like fashion to classify each `noun` as a `subj` or an `obj` according to its position relative to the verb.

```
noun(A) /- verb(_)      ::> subj(A).
verb(_) -\ noun(A)      ::> obj(A).
noun(A), [and], subj(B) ::> subj(A+B).
obj(A), [and], noun(B)  ::> obj(A+B).
```

Context parts perhaps combined with simplification rules can be used for disambiguation of straightforward and otherwise ambiguous grammars. The following shows a grammar for arithmetic expressions with traditional operator precedence; the semicolon denotes alternatives in syntactic context and the `where` notation stands for syntactic replacement (enhances readability only); notice how standard Prolog tests are applied in guards of the two last rules.

```
e(E1),[+],e(E2) /- Rc <:> e(plus(E1,E2))
   where Rc = (['+'];[')'];[eof]).
e(E1),[*],e(E2) /- Rc <:> e(times(E1,E2))
   where Rc = ([*];[+];[')'];[eof]).
e(E1),[^],e(E2) /- [X] <:> X \= ^ | e(exp(E1,E2)).
['('], e(E), [')'] <:> e(E).
[N] <:> integer(N) | e(N).
```

This grammar uses standard 'follow' items but hopefully, it illustrates the flexibility that CHRG provides for a competent grammar writer to engineer compact, yet precise and widely covering languages specifications without too many artificial grammar symbols.

5 Abduction in CHRG

Abduction for language interpretation, which traditionally has required a heavy computational overhead, can be implemented in CHRG in a strikingly simple

way; the principle was first introduced by Christiansen (2002a) and is described in detail in (Christiansen 2005) which also gives background references.

We sketch the principle briefly as follows. Consider the following logical grammar rule in which F refers to a fact about the semantical context for a given discourse; it can be read as a CHRG rule or an equivalent DCG rule.

$$\texttt{a, b, } \{F\} \texttt{ ::> ab} \tag{1}$$

If two subphrases referred to by `a` and `b` have been recognized and the context condition F holds, it is concluded that an `ab` phrase is feasible, grammatically as well as with respect to the context. Analysis with such rules works well when context is known in advance for checking that a given discourse is syntactically and semantically sound with respect to that context.

In case the context is unknown, we have a more difficult abductive problem of finding proper context theory so that an analysis of an observed discourse is possible. Rules of the form (1) are not of much use unless an interpreter that includes abduction is involved.

Our solution is to move the reference to the contextual predicates into the other side of the implication, thus replacing the rule above with the following:

$$\texttt{a, b ::> } \{F\}\texttt{, ab} \tag{2}$$

Intuitively it reads: If suitable `a` and `b` phrases are found, it is feasible to assert F and, thus, under this assumption conclude `ab`.

Obviously, the two formulations (1) and (2) are not logically equivalent but it is straightforward to formulate and prove correctness. For an unambiguous grammar, the transformed version indicated by (2) can be executed as a CHRG with the correct abductive explanation of a discourse being read out of the final constraint store. The full CHRG system includes additional methods for ambiguous grammars to avoid different hypothesis sets for different abductive interpretations to be mixed up.

A specification based on abduction needs *integrity constraints* to suppress senseless explanations. The CHRG system allows to express integrity constraints directly as CHR rules is shown in the following example. An IC is a condition or rule that expresses a restriction

Consider the discourse "*Garfield eats Mickey. Tom eats Jerry. Jerry is mouse. Tom is cat. Mickey is mouse*." We intend to learn from it a categorization of the individuals and which categories that are food items for others. An interesting question is to which category Garfield belongs as this is not mentioned explicitly. The abducible predicates for this grammar are `food_for`/2 and `categ_of`/2, and the following two CHR rules serve as integrity constraints.

```
categ_of(N,C1), categ_of(N,C2) ==> C1=C2.
food_for(C1,C), food_for(C2,C) ==> C1=C2.
```

I.e., the category for a name is unique, and for the sake of this example it is assumed that a given category is the food item for at most one other category. The following part of the grammar classifies the different tokens.

```
[tom] ::> name(tom).    ...
[is]  ::> verb(is).     ...
verb(is) -\ [X] <:> category(X).
```

The last rule applies a syntactic left context part in order to classify any symbol to the right of an occurrence of "`is`" as a `category`.

A sentence "*Tom is cat*" is only faithful to a context if `categ_of(tom,cat)` holds in it. Thus, if sentence "*Tom is cat*" is taken as true, it is feasible to assume `categ_of(tom,cat)`; in general:

```
name(N), verb(is), category(C)
                ::> {categ_of(N,C)}, sentence(is(N,C)).
```

A sentence "*Tom eats Jerry*" is only faithful to a context in which proper `categ_of` and `food_for` facts hold:

```
name(N1), verb(eats), name(N2) ::>
  {categ_of(N1,C1), categ_of(N2,C2), food_for(C1,C2)},
  sentence(eats(N1,N2)).
```

Let us trace the analysis of the sample discourse; only the context facts are recorded. First sentence "*Garfield eats Mickey*" gives rise to

```
categ_of(garfield,X1), categ_of(mickey,X2),
food_for(X1,X2).
```

The "`X`"s are uninstantiated variables. The next "*Tom eats Jerry*" gives

```
categ_of(tom,X3), categ_of(jerry,X4), food_for(X3,X4).
```

"*Jerry is mouse*" gives `categ_of(jerry,mouse)`, and the first IC immediately unifies `X4` with `mouse`. In a similar way "Tom is cat" gives rise to a unification of `X3` with `cat` and `food_for(X3,X4)` has become

```
food_for(cat,mouse).
```

Finally "*Mickey is mouse*" produces `categ_of(mickey,mouse)` that triggers the first integrity constraint unifying `X2` with `mouse` and thus the second IC sets `X1=cat` and there is no other possibility. So as part of the solution to this language interpretation problem, we have found that Garfield is a cat.

6 Conclusion

CHR Grammars founded of current constraint logic technology have been introduced, and their application to aspects of natural language syntax illustrated by small examples. CHRG is a technologically updated ancestor of Definite Clause

Grammars: A relative transparent layer of syntactic sugar over a declarative programming language, providing both conceivable semantics and fairly efficient implementation. In CHRG we have just replaced Prolog by Constraint Handling Rules. The result of this shift is a very powerful formalism in which several linguistic aspects usually considered to be complicated or difficult are included more or less for free:

- Ambiguity and grammatical errors are handled in a straightforward way, all different (partial) parses are evaluated in parallel.
- Context-sensitive rules, which are inherent part of the paradigm, handle aspects of coordination in an immediate way.
- Abduction, which is useful for identifying indirectly implied information, is expressed directly with no additional computational devices needed.

Context-sensitive rules combined with the ability to handle left-recursion (as opposed to DCG) are a great help for producing grammars with relatively few, concise rules without artificial nonterminals; a drawback is the lack of empty production.

There is a large unexplored potentiality in CHRG and language processing by means of CHR. We can mention the possibility of integrating arbitrary constraint solvers, and adding weights to prioritize between different parses (and abductive explanations!) seems quite straightforward. For example, Bistarelli et al. (2002) have shown how to handle soft constraints in CHR and this opens up for integrating recent results in statistically based parsing.

In another paper (Christiansen & Dahl 2002) we have extended with facilities for error detection and correction. Robustness combined with flexibility (e.g., error correction) makes application in speech systems interesting: If, e.g., the phonetic component cannot distinguish a token from being `hats` or `cats`, we simply add both to the input state with identical boundaries. Parsing from a state {`token(0,1,hats)`, `token(0,1,cats)`, `token(1,2,eat)`, `token(2,3,mice)`} will explore the different options in parallel, with only those satisfying syntactic and semantic requirements of the actual grammar leading to a full parse tree. No real-world applications have been developed in CHRG yet, but we have good expectation for scalability as selected grammars can run in linear time. Furthermore, the full flexibility of the underlying CHR and Prolog machinery is available for optimizations. Independently, CHRG is available as powerful modeling and prototyping tool.

In a way, the idea is naïve, almost too naïve, just applying grammar rules bottom-up over and over until the process stops. However, we can rely now on the underlying, well-established computational paradigm of CHR for such rules-based computations.

It is our hope that the availability of the CHRG system can stimulate research in constraint-based language analysis, ideally leading to a full integration of lexical, grammatical, semantical, and pragmatic processing.

Acknowledgements. Part of this work has been carried out while the author visited Simon Fraser University, Canada, partly supported by the Danish Natural Science Council; thanks to Verónica Dahl for helpful discussion and providing a stimulating environment. This research is supported in part by the OntoQuery project funded by the Danish Research Councils, and the IT-University of Copenhagen.

REFERENCES

Abdennadher, Slim & Henning Christiansen. 2000. "An Experimental CLP Platform for Integrity Constraints and Abduction". *Proceedings of Flexible Query Answering Systems: Advances in Soft Computing series* (*FQAS-2000*), *Warsaw, Poland*, 141-152. Heidelberg & New York: Physica-Verlag (Springer).

Abdennadher, Slim & Heribert Schütz. 1998. "CHR$^\vee$: A Flexible Query Language". *Proceedings of International Conference on Flexible Query Answering Systems* (*FQAS-1998*) *Roskilde, Denmark* ed. by Troels Andreasen et al. (= *Lecture Notes in Computer Science* 1495), 1-15. Heidelberg & New York: Springer-Verlag.

Allen, James. 1995. *Natural Language Understanding* (2nd edition). Redwood, Calif.: Benjamin/Cummings.

Bistarelli, Stefano, Thom Frühwirth & Michael Marte. 2002. "Soft Constraint Propagation and Solving in CHRs". *Proceedings of the 2002 ACM Symposium on Applied Computing, Madrid, Spain* ed. by Gary Lamont, 1-5. New York: ACM Press.

Blache, Philippe. 2000. "Constraints, Linguistic Theories and Natural Language Processing". *2nd International Conference on Natural Language Processing* (*NLP-2000*), *Patras, Greece* ed. by Dimitris Christodoulakis (= *Lecture Notes in Computer Science* 1835), 221-232. Heidelberg & New York: Springer-Verlag.

Christiansen, Henning. 2002a. "Abductive Language Interpretation as Bottom-up Deduction". *7th International Workshop on Natural Language Understanding and Logic Programming* (*NLULP'02*) ed. by Shuly Wintner (= *Datalogiske Skrifter* 92), 33-47. Copenhagen, Denmark: Roskilde University, Denmark.

Christiansen, Henning. 2002b. "CHR Grammar web site, Released 2002". — `http://www.ruc.dk/~henning/chrg` [Source checked in May 2004].

Christiansen, Henning. Forthcoming. "CHR Grammars". To appear in *Theory and Practice of Logic Programming*, 2005.

Christiansen, Henning & Veronica Dahl. 2002. "Logic Grammars for Diagnosis and Repair". *International Journal on Artificial Intelligence Tools* 12:3.307-314.

Colmerauer, Alain. 1975. "Les grammaires de metamorphose". *Technical report, Groupe d'Intelligence Artificielle, Université de Marseille-Luminy*. Translated into English as (Colmerauer 1978).

Colmerauer, Alain. 1978. "Metamorphosis Grammars". *Natural Language Communication with Computers* ed. by Leonard Bolc, 133-189. (= *Lecture Notes in Computer Science* 63). Heidelberg & New York: Springer-Verlag. English translation of (Colmerauer 1975).

Dahl, Veronica, Paul Tarau & Renwei Li. 1997. "Assumption Grammars for Processing Natural Language". *Proceedings of the 14th International Conference on Logic Programming, Leuven, Belgium* ed. by Lee Naish, 256-270. Cambridge, Mass.: MIT Press.

Duchier, Denys. 2000. *Constraint Programming For Natural Language Processing. European Summer School on Logic, Language and Information* (*ESSLLI-2000*), Birmingham, England. — `www.cs.bham.ac.uk/research/conferences/esslli/notes/duchier.html` [Source checked in May 2004].

Duchier, Denys & Stefan Thater. 1999. "Parsing with Tree Descriptions: A Constraint-Based Approach". *Proceedings of the 6th International Workshop on Natural Language Understanding and Logic Programming* (*NLULP'99*), 17-32. Las Cruces, New Mexico. — `www.neci.nec.com/homepages/sandiway/nlulp99/toc.html` [Source checked in May 2004].

Frühwirth, Thom. 1998. "Theory and Practice of Constraint Handling Rules". *Constraint Logic Programming* (= Special Issue of *Journal of Logic Programming*) 37:1/3.95-138.

Hecksher, Thomas, Sune T. B. Nielsen & Alexis Pigeon. 2002. *A CHRG Model of the Ancient Egyptian Grammar*. Unpublished student project report, Computer Science Dept., Roskilde University, Denmark.

Maruyama, Hiroshi. 1990. "Structural Disambiguation with Constraint Propagation". *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics* (*ACL'90*), 31-38. Univ. of Pittsburgh, Pittsburgh, Penn.

Meyer, Bernd. 2000. "A Constraint-based Framework for Diagrammatical Reasoning". *Journal of Applied Artificial Intelligence* 14:4.327-244.

Morawietz, Frank. 2000. "Chart Parsing as Contraint Propagation". *Proceedings of the 18th International Conference on Computational Linguistics* (*COLING-2000*), vol. 1, 551-557. Saarbrücken, Germany.

Pereira, Fernando C. N. & David H. D. Warren. 1980. "Definite Clause Grammars for Language Analysis — A Survey of the Formalism and a Comparison with Augmented Transition Networks". *Artificial Intelligence* 13:3.231-278.

Schröder, I., W. Menzel, K. Foth & M. Schulz. 2000. "Dependency Modelling with Restricted Constraints". *International Journal Traitement Automatique des Langues: Les grammaires de dépendance* 41:1.113-144.

Shieber, Stuart M., Yves Schabes & Fernando C. N. Pereira. 1995. "Principles and Implementation of Deductive Parsing". *Journal of Logic Programming* 24:1/2.3-36.

Swedish Institute of Computer Science. 2003. *SICStus Prolog User's Manual, Version 3.10*. — `http://www.sics.se/isl` [Source checked in May 2004].

Using Parallel Texts to Improve Recall in Botany

MARY MCGEE WOOD*, SUSANNAH J. LYDON*, VALENTIN TABLAN,**
DIANA MAYNARD** & HAMISH CUNNINGHAM**

*University of Manchester
**University of Sheffield

Abstract

Applying known IE techniques to independent parallel texts describing the same information and merging the results brings significant improvements in performance. Recall summed over six botanical descriptions of several plant species is triple the average for each text individually. We analyse these results, and describe the processing techniques used. This leads us to the concept of "greedy extraction", where the information template is not frozen before processing, but can incorporate new information found in the text(s).

1 Overview – parallel texts and greedy extraction

Our experiments show that applying known IE techniques to independent parallel texts describing the same information, and merging the results, brings significant improvements in performance. Recall summed over six botanical descriptions of several plant species is more than triple the average for each text individually. This arises from two independent factors:

(1) The source texts contain different subsets of the total desired information space;
(2) Failures in extraction from any one text are significantly often corrected by accurate results from other texts. 50% of all false negatives are compensated by results from other sources; in 24% of template slots, a false negative in processing one text is compensated by accurate processing of one or more others.

The use of redundancy, in the form of analysing multiple parallel sources, has been explored in computational lexicography (Veronis & Ide 1991) and in Information Retrieval (Ingwersen 1996), with promising results. Veronis & Ide derived concept hierarchies ("ontologies") from five machine readable dictionaries using keywords in definition texts. Since these definitions are informal and at times inaccurate, the individual ontologies were only 30-45% complete and correct compared to human judgement. However, correlating and merging them yielded a 94% success rate. Missing classifications were filled in (recall), and mis-classifications were corrected (precision), by correlation with accurate analyses of other sources. Exploiting redundancy across multiple sources will have

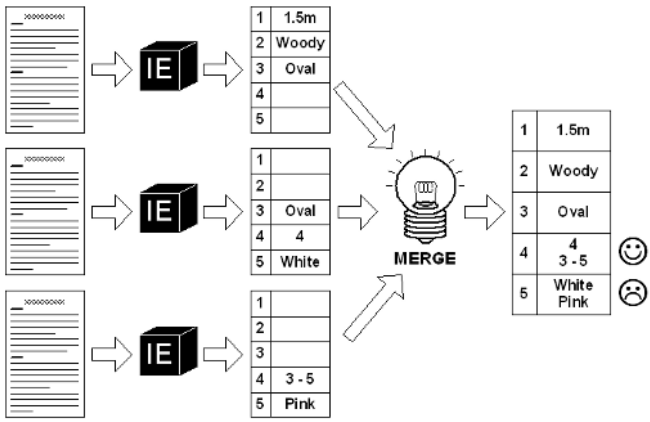

Figure 1: IE *from parallel texts*

different effects in detail in different domains: in the present case, it is primarily recall which is dramatically improved.

Traditionally, IE is restricted to finding answers to a predetermined set of questions encoded as the slots in a template. Any information in the source text which is not mapped to a slot in the template will not be extracted. This is appropriate for many purposes, but ours is different: we do not want to limit in advance what information can be extracted from our source texts, but fully to exploit the information resources they offer. We call this "greedy" extraction. This has significant implications for the techniques to be used. We cannot rely on designated keywords, for example, to help fill a template. Shallow parsing techniques are used to identify heads and their associated modifiers. The initial template is hand-built by a domain expert, informed by the texts. New heads can be added to the template as they are found, through an interactive process which supports - but does not replace - the domain expert.

Systematics(the science of identifying, describing, naming and classifying, and tracing the origins of living things), like all mature descriptive sciences, is rich in written data, and particularly in semi-formal natural language records of careful observations repeated across time and space. The need to make these important primary resources for biodiversity research available in structured electronic form is urgent, as recognised by G7 and the Biodiversity Convention, among others.

Botanical description offers multiple parallel independent descriptions of the same information (descriptions of plant groups); a well understood technical domain suited to IE; and a genuine real-world need for language engineering sup-

port. Botanical texts (Floras) are variable in information content — different authors' descriptions of the same plant group describe somewhat different features. Thus a matrixof data points against sources is sparsely populated (cf. Table 1). Floras also differ in vocabulary and phraseology.

2 Parallel texts – corpus and data

Our training set comprises the descriptions of five species of Ranunculus (buttercup) given in six Floras, chosen to be both representative of the many available, and independent of each other. Together with the higher-order taxon (genus and family) descriptions, this gave us 42 texts of average length 116 words (range 32-208). Although short by IE standards, this is typical of the text-type. The texts were scanned and the OCR output hand-corrected: on the scale of our experiments, the time involved was acceptable. In our target domains for expansion, the sources are already in electronic form, so we do not see this as an obstacle to scaling up. The test set comprises descriptions of five species of the Lamiaceae (mints and related herbs). The text-type is syntactically similar to the training set, while the plant morphology and associated vocabulary are sufficiently different to test the scalability of our approach.

Most botanical descriptions are not true "natural" language, but highly stylised lists of noun phrases, usually lacking verbs and articles altogether. This leads to an information-dense text type (a result of the space limitations historically imposed on authors of Floras). Here is a short, but otherwise typical example:

> 1. R. acris L. - Meadow Buttercup. Erect perennial to 1m; basal leaves deeply palmately lobed, pubescent; flowers 15-25mm across; sepals not reflexed; achenes 2-3.5mm, glabrous, smooth, with short hooked beak; 2n=14. (Stace 1997)

Although problematic for a general-purpose parser, this text-type is highly amenable to analysis with specifically tuned resources. These are small datasets, by normal IE standards, but adequate to provide proof of concept. Work is in progress on scaling up.

Based on hand analysis of the texts, we built a three-dimensional data matrix for each species, where columns represent Floras, rows represent characters, and the third dimension is the taxonomic hierarchy, allowing inheritance of information from higher taxa into species descriptions where appropriate. Table 1 shows a small section of one of these matrices.

The matrix is sparsely populated: it is common for only one of the six Floras to give any value for a character. Thus, for example, the six descriptions of R. acris have, on average, 44 slots each in their individual templates (range 17-68). The combined template for R acris from all sources has 134 slots. Frequently one Flora gives a single value or narrow range for a character while another gives a

	CTM	FF	FNA	GLEASON	GRAY	STACE
stem	stem		Stems	Stems	Stems	
size	10-30cm					
orientation	creeping (on mud or the upper part floating			creeping	closely creeping	
branching	branched					
form					succulent	
surface			glabrous			

Table 1: *Part of a typical data matrix*

wider range (“3”/ “3(-5)”/ “3-5(-7)”): here some simple reasoning is needed to confirm that the various values are compatible. Variations in terminology are also common (“3-lobed”/ “3-parted”/ “tripartite”; “stem leaves”/ “cauline leaves”), and can be accommodated with an adequate glossary or lexicon. Certain alternants in phraseology also recur (e.g., “lobed”/ “with lobes”), and can easily be handled in the grammar. Genuine disagreements are very rare. (See Lydon et al (2003) for further detail.)

The nature of the source texts thus limits the potential recall per text against the total sum of available information, and forms in itself an strong argument for the analysis of parallel texts. However also, as we will see, where texts do contain the same information, failures in extraction from one text are significantly often compensated by successes from others.

3 Shallow parsing

Our system builds on the Natural Language Processing capabilities provided by GATE, a General Architecture for Text Engineering, developed by the NLP group at the University of Sheffield (Cunningham et al 2002); it is based on ANNIE, an information extraction system integrated into GATE. The system comprises a version of ANNIE’s main processing resources: tokeniser, sentence splitter, POS tagger, gazetteer and finite state transduction grammar. The resources communicate via GATE’s annotation API, which is a directed graph of arcs bearing arbitrary feature/value data, and nodes rooting this data into document content (in this case text).

IE technology is used to recognise and characterise information about plants and their features. This enables the recognition of information such as text headers, details of plants, recognition of their components and subcomponents (such as leaves, stems, root and descriptions of their characteristics (such as the size of the plant, leaf shape, petal colour, etc.). This is achieved using the resources described above to annotate textual patterns (using text strings, punctuation, syntactic information etc. as evidence). Relations can then be identified between

identified patterns so that features can be correctly linked to the relevant component.

The JAPE (Java Annotation Pattern Engine) grammars identify patterns such as Modifier + Head, e.g., "short caudex", where the Modifier consists of an adjectival group, and the Head (a plant component) is taken from a gazetteer list. Heads are annotated as *Head*, while modifiers are annotated as *PlantFeature*. Text headers are also annotated as *Header*, e.g., "1. R. acris L. - Meadow Buttercup." in the text above. Each annotation may be given optional attributes, which are used either to further classify the heads and features (and to place them correctly in the database at export time), or simply for debugging purposes (e.g., to show which rule has been fired). In the example above, "short" is annotated as a *PlantFeature* and given the attribute *AdjHead* (because it precedes the Head), while "caudex" is annotated simply as *Head*.

The last processing resource used is a results extractor which parses the previously generated annotations and associates the sets of features with their respective heads, using the information previously described (such as *HeadAdj* and *AdjHead*) to determine the correct association. It also extracts the section header information from the input texts. Once all the information is collected, it is exported in a text file in tab-separated-values format that can be used with other tools (e.g., MS Excel) for results analysis.

The system output for Stace's description of R. acris (given above) is shown in Table 2. The only significant mistake is that the achene-beak has been overlooked. Compared to the gold standard template for this one text, recall is 82%, precision is 93%.

Header: 1. R. acris L. - Meadow Buttercup

HEAD	KIND	FEATURE	TYPE	KIND	NEGATION
		Erect perennial			
		to 1m	measure	unknown	
		2n=14	count	chromosome	
basal leaves	position Prefix	pubescent			
		deeply palmately lobed			
flowers		15-25mm across	measure	width	
sepals		reflexed			true
achenes		short hooked			
		smooth			
		glabrous			
		2-3.5mm	measure	unknown	

Table 2: *System results for R. acris in Stace 1997*

4 Evaluation of information merging

Evaluation of our results must allow for the fact that each text in a set (of descriptions of the same species) contains only a subset of the total sum of information

in all texts, which is our target and gold-standard. We must therefore evaluate the performance on each text twice: against a text-specific template containing slots for exactly the information which is actually in that text, and against the total.

The figures for R. acris are typical. The text-specific template size averages 44 slots, compared with 134 in the combined template: a range from 27% to 62% of the total, averaging 39%. The percentage figures from the text-specific evaluation, multiplied by the appropriate percentages of the whole template, give us the figures for each text against the total.

The recall figures for the six descriptions of R. acris in the intial dataset, each compared with its own text-specific template, range from 51% to 85%, average 70%. For each against the combined template, the figures are 10% - 34%, average 22%. The example above is our shortest text, and the most accurately analysed; however, its individual template contains only 13% of the slots in the complete template for the species, so recall by that standard is only 10%.

Table 3 shows the system's performance for the first three sets of texts to be analsysed in detail. The most significant figures are highlighted: the average recall from a single text, compared to the full template, is 22%, while the recall of the six combined is 71%. 50% of all instances of false negatives are compensated for by one or more other sources. Compensation for false negatives occurred in 24% Â**?**of the total number of slots in the full template (including the missing achene-beak in the example above).

Precision currently averages 78% against single templates (again better for shorter texts), 63% against the whole template. We are able to use a conservative account of precision, as recall from parallel sources compensates for a high proportion of false negatives, so we are satisfied with a relatively low figure here. The performance numbers for the individual texts against the combined template are low by current IE standards, precisely because the template is so large. Also, even the hand-built gold-standard data matrix is sparsely populated: some of the performance gain from merging results across texts is inherent in the texts themselves. However, our results show that correlating extraction results across texts also, with significant frequency, compensates for processing errors.

5 Greedy extraction

As discussed earlier, our concept of greedy extraction differs from traditional information extraction in that we do not determine in advance a closed set of slots in the template which we wish to fill. Rather, we aim to be able to grow or adapt the template when new text sources bring in new types of information. This may happen on a small scale when a new text is analysed describing a known species. It will be a significant issue when a new species is considered for the first time, possibly with parts or features not found in previous analyses. For example, in extending our system coverage from the training set (Ranunculus) to the test set

	R. acris	R. bulbosus	R. hederaceus	Mean
True positives across all descriptions	176	146	127	150
False negatives across all descriptions (A)	90	125	44	86
False positives across all descriptions	55	61	30	49
Average single description recall	70	55	74	66
Average single description precision	78	60	83	74
Avg single description recall for whole template	22	18	26	**22**
Avg single description precision for whole template	78	60	83	74
true positives from merged results	92	85	68	82
false negatives from merged results	42	54	15	37
Total number of slots in whole template (B)	134	139	83	119
Merged recall for whole template	69	61	82	**71**
Merged precision for whole template using total false positives	63	58	69	63
sum of all instances of false negatives compensated for by other sources (C)	45	57	24	42
Of all instances of missed information, **percentage compensated for by merging** (C/A*100)	50	46	55	**50**
number of slots where false negatives compensated for by other sources (D)	34	41	15	30
Of total number of slots in template, **percentage where merging allowed compensation** for missed information (D/B*100)	25	29	18	**24**

Table 3: *Accuracy of processing for six descriptions of three species*

(Lamiaceae), the main "plant parts" and characters described were the same for the two groups. Proceedings of the The sub-parts, however, did vary, mainly in the flowers: floral structure differs significantly in the mints from the simple buttercup, so new terms and features were introduced. (There are around 110 slots for inflorescence in Ranunculaceae, around 190 in Lamiaceae).

Our method for greedy extraction builds on work at Sheffield on the HaSIE system (Maynard et al 2002), which detected information about health and safety issues from company reports. Here the set of slots was initially unknown, and generated partly by manual analysis of the texts, and partly by system-generated output. It was fixed after system training, and could not be modified automatically.

Our approach takes this a logical step further: the set of slots can be modified by the user in an interactive process which is interleaved with parsing. Unfamiliar elements detected by the shallow parser are presented to the user. New concepts can be added to the gazeteer lists — thus fused petals, unknown in Ranunculus, are typical of the Lamiaceae. New vocabulary - synonyms for known concepts, as when one author calls petals "honey-leaves" — can be added to the lexicon. (We expect this to be relatively rare, as the system already includes a comprehensive botanical glossary derived from Lawrence (1951).) This enables us to pick up new types of information found in new text sources, which would

be ignored by traditional methods, while maintaining the formal integrity and boundedness of the knowledge representation.

Looking at the range of significant information found even in closely related, largely consensual technical text in a small domain, it becomes clear that we stand to gain significantly if we can adopt an open-ended, "greedy" approach to Information Extraction. The very success of this approach leads to a problem of scale. Our first template for five species of Ranunculus described by six authors was eventually simplified to 271 slots: e.g.,

```
<FRUIT_BEAK>:=
FRUIT_BEAK_PRESENCE:    "PRESENCE"
FRUIT_BEAK_LENGTH:      "DISTANCE"
FRUIT_BEAK_SHAPE:       "SHAPE"
FRUIT_BEAK_SURFACE:     "TEXTURE"
FRUIT_BEAK_POSITION:    "POSITION
FRUIT_BEAK_ORIENTATION: "ORIENTATION"
FRUIT_BEAK_DEFINITION:  "DEFINITION"
FRUIT_BEAK_TIP:         <FRUIT_BEAK_TIP>
```

where <FRUIT_BEAK> is part of an entity <FRUIT> which is part of the whole plant, and <FRUIT_BEAK_TIP> is an entity which is part of <FRUIT_BEAK> and can have shape, orientation etc in its own right.

Our eventual aim is an architecture in which a single, independent domain-specific information lattice, built in a Description Logic (Horrocks 2002), is accessed by all relevant processing modules and also serves as the "template" for placing extracted information. This one component can then be replaced for new domains.

6 Prospects

If Systematics appears "narrow", other important technical domains also have data unhelpfully distributed over multiple text sources. In Bioinformatics, for example, new research results are appearing rapidly in journals and in the comment fields of various large databases (e.g., in 2000, 130 publications per month on E. coli metabolism). Keeping up with these developments currently requires massive human effort. SWISS-PROT is probably the largest, most accurate, and most important of the databases in this field:

> SWISS-PROT ... depends on a vast human network to generate its annotation by systematically running analysis software, reading the literature, contacting experts in the field, and so on. Clearly, there is a demand for information that helps the consumer turn data into understanding ... the challenge is to make this kind of human effort sustainable. (Miller & Attwood 2003)

This domain shares with botany all the characteristics which support our multiple-source, greedy techniques. Our interactive approach to template expansion is also ideally suited to "mak[ing] this kind of human effort sustainable": supporting, not replacing, the domain expert is central to our long-term objectives. The volumes of text to be processed in this domain are larger than in botany, but they already electronic, so our only real bottleneck, hand correction of OCR output, is not an issue. Beyond the natural sciences, it may prove fruitful to apply our technique to parallel multilingual texts, either naturally occurring, or obtained by machine translation.[1]

Traditional work in IE has drawn on intuition about a small core of important common information in domain-specific texts to build a predetermined template. HaSIE brought a first degree of automation to the task of template building *from* texts. Parallel texts offer a significant increase in the total amount of information available to be extracted in a domain. Greedy extraction makes the most of this. The improvement we have shown in actual extraction from 22% to 71% proves the value of analysing parallel text descriptions to extract the greatest possible amount of useful information.

Acknowledgements. The MultiFlora project is funded by the BBSRC / EPSRC Joint Bioinformatics Initiative, grant 34/BIO12072, and BBSRC Bioinformatics and E-science Initiative, grant 34/BEP17049.

REFERENCES

Cunningham, Hamish, Diana Maynard, Kalina Bontcheva & Valentin Tablan. 2002. "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications". *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (*ACL'02*), 168-175. Philadelphia, U.S.A.

Horrocks, Ian. 2002. "An Ontology Language for the Semantic Web". *IEEE Intelligent Systems*, 74-75.

Ingwersen, P. 1996. "Cognitive Perspectives of Information Retrieval Interaction: Elements of a Cognitive IR Theory". *Journal of Documentation* 52:1.3-50.

Lawrence, G.M.H. 1951. *Taxonomy of Vascular Plants*. Macmillan, New York.

Lydon, Susannah.J., Mary McGee Wood, Robert Huxley & David Sutton. 2003. "Data Patterns in Multiple Botanical Descriptions: Implications for Automatic Processing of Legacy Data". *Systematics and Biodiversity* 1:2.151-157.

Maynard, Diana, Kalina Bontcheva, Horacio Saggion, Hamish Cunningham, & Oana Hamza. 2002. "Using a Text Engineering Framework to Build an Extendable and Portable IE-based Summarisation System". *Proceedings of the Association for Computational Linguistics Workshop on Text Summarisation*, 19-26, Philadelphia, U.S.A.

[1] We are grateful to an anonymous referee for this suggestion.

Miller, Crispin J. & Terri K. Attwood. 2003. “Bioinformatics Goes back to the Future”. *Nature Reviews Molecular Cell Biology* 4:157-162.

Veronis, Jean & Nancy Ide. 1991. “An Assessment of Semantic Information Automatically Extracted from Machine Readable Dictionaries”. *Proceedings of the European Chapter of the Association for Computational Linguistics*, 227-232. Berlin.

Marking Atomic Events in Sets of Related Texts

ELENA FILATOVA & VASILEIOS HATZIVASSILOGLOU
Columbia University

Abstract

The notion of an *event* has been widely used in the computational linguistics literature as well as in information retrieval and various NLP applications, although with significant variance in what exactly an event is. We describe an empirical study aimed at developing an operational definition of an event at the *atomic* (sentence or predicate) level, and use our observations to create a system for detecting and prioritizing the atomic events described in a collection of documents. We report results from testing our system on several sets of related texts, including human assessments of the system's output and a comparison with information extraction techniques.

1 Introduction

Events have received a lot of attention in the theoretical linguistics literature (e.g., Chung & Timberlake 1985, Bach 1986, Pustejovsky 2000). At the same time, several natural language applications deal with information about events extracted from a text or collection of texts, for example, information retrieval systems participating in the Topic Detection and Tracking initiative (Yang et al. 1999; Allan 2002), and information extraction systems participating in the Message Understanding Conferences (Marsh & Perzanowski 1997). But however intuitive the notion of 'event' might seem, exact definitions of what an event is are usually not supplied, and the implicit definitions seem to conflict. Not only is the exact meaning of an event in dispute, but also the extent of an event's realization in text.

Linguists have been looking at semantic constraints in sentences to distinguish between events, extended events, and states; see for example (Chung & Timberlake 1985, Bach 1986, Pustejovsky 2000). Often in such research the event analysis is based on properties of the verb, and verbs are classified according to their relationships with event classes (Levin 1993).

From a computational perspective, there have been attempts to classify verbs into those denoting events and those denoting processes (Siegel & McKeown 2000) and to investigate what text structure can be considered to be a minimal unit for event description (McCoy & Strube 1999, Filatova & Hovy 2001). The work most commonly referred to as event detection is that originating from the Topic Detection and Tracking (TDT) research effort sponsored by DARPA. An important contribution of that research program is the recognition of the distinction between an event and a topic.

Systems participating in the Scenario Template task of the Message Understanding Conference (Marsh & Perzanowski 1997) competitions use information extracted and inferred from a text to fill in the appropriate fields in predefined templates corresponding to the domain of the text. However, the MUC systems suffer from two drawbacks: First, the fixed templates preclude detecting multiple events of different types, or of types not anticipated during system design. Second, they are heavily dependent on the domain, which requires a lot of time to create accurate templates defining possible events for that particular domain, and even more effort in adapting the system to the sublanguage and knowledge model of that domain.

In this paper, we draw on a synthesis of the above three competing approaches to events (linguistics, information retrieval, and information extraction) to obtain a method for constructing a representation of the *atomic events* in multiple related documents. We aim at small text pieces and multiple low-level events rather than the most generic events targeted by IR, but we incorporate information about the similarity of texts to find topic-specific relationships. We do not rely on predefined templates and slots, as IE does, but we discover relationships in a domain-independent fashion and label them with appropriate verbs and nouns. Our approach is informed by linguistic theory, but remains operational for arbitrary texts.

2 A study of event annotation

As it has been noted in the introduction there is huge diversity in both the structure and length of events. Thus, before going any further we describe a two-stage evaluation experiment the major goal of which is to obtain a definition of events that can be used in a system for the automatic detection and extraction of events from a document.

We conducted our first study of text annotation for event information by asking a number of computer science graduate students (mostly in computational linguistics) to mark text passages that describe important events in news stories. The annotators were given 13 news articles randomly selected from the DUC-2001 (Document Understanding Conference) corpus. The texts varied in length from 15 to 60 sentences. Five of the thirteen texts were each annotated by two participants in the study.

We deliberately provided no definition of *event* for this study, to see if the respondents would naturally converge to an operational definition (as evidenced by high agreement on what they marked). Our study had two further aims: to determine what text range in the absence of instructions on the length of what they should mark, people tend to favor as the appropriate text parts describing an event; and to gather evidence of features that frequently occur in the marked passages and could be automatically extracted by a system simulating the human

annotators.

We noticed substantial disagreement between annotators on what text passages should be marked as events. Since our annotation instructions left unspecified the length of event descriptions, a basic text unit that could be marked or unmarked is not defined either and therefore it is hard to quantitatively measure annotators' agreement.

While the annotators disagreed on what text pieces to select as event descriptions, they exhibited more agreement on how long these pieces should be. Out of 190 text regions marked as events, 46 (24%) consisted of one clause within a longer sentence, 22 (11%) of one sentence minus one short prepositional phrase, 95 (50%) of exactly one sentence, and 27 (14%) of multiple sentences.[1]

We analyzed the passages marked as event descriptions looking for text features that could be included in an automated event detection system. Naturally, the verb itself often provides important information (via tense, aspect, and lexical properties) about the event status of a clause or sentence. In addition, the following features are correlated with the presence of events: *Proper nouns* occur more often within event regions, possibly because they denote the participants in events. In contrast, *pronouns* are less likely to occur in event regions than in non-events. As expected, the presence of *time phrases* increases the likelihood of a text region being marked as an event.

We thus came up with a procedural definition of atomic events which we used for detecting, extracting and labeling of atomic events in our system. The details of this definition are presented in the next section.

3 Detecting and labeling events

Drawing from our event annotation study, we decided on an algorithm for detecting, extracting, and labeling events that is based on the features that seemed more strongly correlated with event regions. Event regions are contained within a sentence. Thus, we anchor events on their major constituent parts (Named Entities for people and organizations, locations, and time information)[2] and expect at least two such major elements in a sentence to consider extracting an event. The procedure for extracting atomic events is the following:

- We analyze a collection of documents clustered on a specific topic.
- We take the sentence as the scope of an event. Our algorithm ignores sentences that contain one named entity or none.

[1] Although words like *war* or *earthquake* can denote events, single nouns were never marked as events by our annotators.

[2] All these major elements can be retrieved with a named entity tagger; we use BBN's IdentiFinder (Bikel et al. 1999).

- We extract all the possible pairs of named entities (preserving the order). Such pairs of named entities are called *relations*.
- For each relation we extract all the words that occur in-between the elements of the relation. These are extracted together with their part of speech tags which we get with the help of Collins' (1996) parser.
- Out of all the words that occur in-between elements of relations we are now interested only in those which are either non-auxiliary verbs or nouns which are hyponyms of *event* or *activity* in WordNet (Miller et al. 1999). We call these words *connectors*.
- For each relation we calculate how many times it occurs, irrespective of the connectors.
- For each connector we calculate how many times this connector is used in a particular relation.

Our hypothesis is that if named entities are often mentioned together, these named entities are strongly related to each other within the topic from which the relation was extracted. Although our method can be applied to a single text (which by itself assures some topical coherence), we have found it beneficial to extract events from sets of related articles. Such sets can be created by clustering texts according to topical similarity, or as the output of an information retrieval search on a given topic. Following the above procedure we create a list of relations together with their connectors for a set of documents.

Out of all the relations we leave only the top n events that have the highest frequency out of all the relations and we also eliminate those relations that are not supported by high frequency connectors (both of these parameters are adjustable and are determined empirically).

We then examine the graph of connections induced by the surviving pairs. For each two edges in that graph with a common endpoint (e.g., (A, B) and (A, C)), we examine if their list of connectors is substantially similar. We consider two such lists substantially similar if one contains at least 75% of the elements in the other. When that condition applies, we merge the two candidate events into one link between A and a new element $\{B, C\}$ (i.e., we consider B and C identical for the purpose of their relationship to A), and add the scores of the two original events to obtain the score of the composite event.

4 System output

We ran our system on a subset of the topics provided by the Topic Detection and Tracking Phase 2 research effort (Fiscus et al. 1999). The topics consist of articles or transcripts from newswire, television, and radio. We used 70 of the 100 topics, those containing more than 5 but less than 500 texts. Since human annotators created these topical clusters in a NIST-sponsored effort, we can be assured of a certain level of coherence in each topic.

Relation Frequency	First Element	Second Element
0.0212	China Airlines	Taiwan
0.0191	China Airlines	Taipei
0.0170	China Airlines	Monday
0.0170	Taiwan	Monday
0.0170	Bali	Taipei
0.0148	Taipei	Taiwan

Table 1: *Top 6 named entity pairs for the 'China Airlines crash' topic*

Relation	Connector	Connector Frequency
China Airlines – Taiwan	crashed/VBD	0.0312
	trying/VBG	0.0312
	burst/VBP	0.0267
	land/VB	0.0267
China Airlines – Taipei	burst/VBP	0.0331
	crashed/VBD	0.0331
	crashed/VBN	0.0198
Taipei – Taiwan	–	–

Table 2: *Top connectors for three of the relations in Table 1*

TDT provides descriptions of each topic that annotators use to select appropriate documents by issuing and modifying IR queries. Here is the official explication of one topic ("China Airlines crash"):

> *The flight was from Bali to Taipei. It crashed several yards short of the runway and all 196 on board were believed dead. China Airlines had an already sketchy safety record. This crash also killed many people who lived in the residential neighborhood where the plane hit the ground. Stories on topic include any investigation into the accident, stories about the victims/their families/the survivors. Also on topic are stories about the ramifications for the airline.*

Table 1 shows the top 6 pairs of named entities extracted from the topic at the first stage of our algorithm (before considering connectors). The normalized relation frequency is calculated by dividing the score of the current relation (how many times we see the relation within a sentence in the topic) by the overall frequency of all relations within this topic.

It is clear from the table that the top relations mention the airline company whose plane crashed (*China Airlines*), where the crash happened (*Taiwan*, *Taipei*), where the plane was flying from (*Bali*), and when the crash happened (*Monday*). Interestingly, we obtain a clique for the three elements *China Airlines*, *Taiwan* and *Taipei*. Let us analyze the connectors for the three pairs among these three elements (Table 2). The normalized connector frequency is

calculated by dividing the frequency of the current connector (how many times we see this connector for the current relation) by the overall frequency of all connectors for the current relation. According to this table the relation *Taiwan – Taipei* does not have any connectors linking these two named entities. For us it means that the named entities in the relation *Taiwan – Taipei* are linked to each other not through an event but through some other type of static relation (indeed, Taipei is the capital of Taiwan).

Finally, we factor in topic specificityfor the extracted events. We calculate the ratio of the relation frequency for a specific topic over the relation frequency of that same pair for all the topics under analysis. This ratio is equal to *1.0* for the relations *China Airlines – Monday, Bali – Taipei* and it is equal to *0.0850* for *CNN – New York*. The specificity feature is helpful in deciding what relations are important for a given topic and what are not.

We close this section with a comment on the anchor points used by our algorithm. Such anchor points (by default named entities) are necessary in order to limit the amount of relations considered. We chose named entities on the basis of our analysis of events marked by people (Section 2). However, the system is adaptable and the user can specify additional words or phrases that should be used as anchor points. In this example, if the word *passengers* is submitted to the system, then the third most important event extracted will refer to the deaths of the passengers. The problem of how to find the best candidate nouns to add to the list of nouns which anchor events (by default this list consists only of named entities) is very similar to the problem of creating a right query within Information Retrieval tasks.

5 Comparison with Information Extraction

We compare our system's output to ideal output for one of the most well-known information extraction competitions, the Message Understanding Conferences (Marsh & Perzanowski 1997) organized by NIST between 1992 and 1998. In MUC's Scenario Template task events are extracted for several pre-specified domains. For each domain a list of templates is created in advance and event extraction is equated to filling these templates, a typical approach in information extraction. Events are extracted from one text at a time and not a collection of texts. [3]. Each text can contain one, several, or no events.

MUC systems produce output in the form of predefined well-structured templates. Classical IE systems have developed a repository of powerful tools intended for extracting information within specific domains. Our system based on the presented definition of atomic events is domain-independent; though it does

[3] There are IE systems which try to fill predefined templates from several texts but during the MUC competition systems analyzed and extracted events for one text at a time.

not assign slots to the constituent parts of the extracted events, our system goes beyond the limits of predefined templates and extracts all possible relations it can find. For example, MUC systems are not supposed to extract any events for the following MUC-7 text:

> *Paris (France), 5 April 90 (AFR) – Colombian leader Virgilio Barco briefed French president François Mitterand here Wednesday on the efforts made by Bogota to fight the country's powerful cocaine traffickers. Mr. Barco told reporters after the meeting at the Elysée Palace that the French leader, who visited Bogota in October 1989, had said once again that he was "very interested" in the drug problem.*

This text is from the terrorism domain collection. And though really no terrorist attacks are described in this text it does not mean that there are no events described. These events include the meeting between François Mitterand and Virgilio Barco, Mitterand's earlier visit to Bogota, and Barco's speaking to reporters. Thus, following the described procedural definition, our system extracts information about this meeting by pointing out that the named entities in the relations *François Mitterand – Virgilio Barco*, *François Mitterand – Wednesday* and *Virgilio Barco – Wednesday* are all linked to each other through the connector *briefed/VB*.

In fairness to the MUC systems we note that they perform additional tasks such as the semantic classification of the information (deciding which slot to select for a given piece of extracted text). Our approach does not assign labels such as *perpetrator* or *target* to named entities. It provides for a more superficial "understanding" of the elements of the event and the roles they play in it, in exchange for portability, generality and robustness.

6 System evaluation

6.1 *Methodology*

To evaluate our system we chose randomly 9 topics out of the 70 TDT-2 topics containing more than 5 and less than 500 texts. For each of these topics we randomly chose 10 texts, ran our system on these 10 texts only, and produced a ranked list of events with verb and noun labels, as described above. Each set of ten texts, and the top ten events extracted by our system from it, were given to one evaluator. The evaluators were asked to first read the texts[4] and then provide a numerical score for the system in the following areas:

- Whether the named entities in the events extracted by our system are really related to each other in the texts. A separate score between 0 and 1 was given for each extracted event.

[4] Which was the reason we limited the number of texts per topic to 10.

- Whether the extracted relations between named entities, if valid, are also important. Again a 0 to 1 score was assigned to each extracted event.
- Whether the label(s) provided for a (valid) event adequately describe the relationship between the named entities.

For these three questions, the evaluators gave a separate score for each extracted event. They were free to use a scale of their own choosing between 0 (utter failure) and 1 (complete success).

6.2 *Results*

Table 3 shows the scores obtained during the evaluation. We report the average rating our system obtained on each of the three questions, across both the ten extracted events in each set and the nine evaluators/topics. We also report the percentage of extracted events that received a non-zero score and a score above 0.5.

Question	Avg. rating	% non-zero	% above 0.5
Link quality	0.7506	92.22%	74.44%
Importance	0.6793	95.00%	62.87%
Label quality	0.6178	90.91%	51.09%

Table 3: *Evaluation scores for our system*

We note that the easiest task for the system is to find valid relationships between named entities, where we obtain about 75% precision by either the average score or the number of scores above 0.5. Next comes the task of selecting important links, with precision of 63–68%.[5] The hardest task is to provide meaningful labels for the events; we succeed in this task slightly in more than half of the valid extracted events, or approximately 40% of the total extracted events. Our system is getting lower scores for topics where the event's major constituent parts are not represented by a named entity (i.e., an earthquake topic) or for very disperse topics, (i.e., the topic about the Israeli-Palestinian peace negotiations.) Regardless, our system overall extracted at least somewhat useful information, as manifested by the fact that over 90% of the reported events received non-zero scores.

7 Conclusions

We have reported on an empirical study of event annotation and a new approach for automatic event detection in text. We have implemented a robust, statistical

[5] Importance and label quality are measured only on extracted relations of reasonable quality (with link quality score above 0.5, 74.44% of the total extracted events).

system that detects, extracts, and labels atomic events at the sentence level without using any prior world or lexical knowledge. Our system uses text collections vs. individual texts (cf. IR approach). It extracts relations between named entities and links that relate these named entities (cf. IE approach), though, in contrast to the classical IE task our relations and links are domain-independent and not defined beforehand but built on the fly. To assign labels to the extracted relations our system uses both verbs and nouns (cf. computational linguistics). The system is immediately portable to new domains, and utilizes information present in similar documents to automatically prioritize events that are specific (and therefore likely more interesting) to a given set of documents. Our examination of results and a first small-scale evaluation indicate that the approach is promising as a means for obtaining a shallow interpretation of event participants and their relationships.

The extracted event information can be used for indexing, visualization and question-answering.

Acknowledgments. We wish to thank Kathy McKeown and Becky Passonneau for numerous comments and suggestions on earlier versions of our system, and John Chen for providing tools for preprocessing and assigning parts of speech to the text. We also thank the members of the Natural Language Group and other graduate students at Columbia University who participated in our evaluation experiments. This work was supported by ARDA under Advanced Question Answering for Intelligence (AQUAINT) project MDA908-02-C-0008. Any opinions, findings, or recommendations are those of the authors and do not necessarily reflect ARDA's views.

REFERENCES

Allan, James, ed. 2002. *Topic Detection and Tracking: Event-Based Information Organization.* Boston, Mass.: Kluwer Academic.

Bach, Emmon. 1986. "The Algebra of Events". *Linguistics and Philosophy* 9:1.5-16.

Bikel, Daniel M., Richard Schwartz & Ralph Weischedel. 1999. "An Algorithm that Learns What's in a Name". *Machine Learning* 34:1/3.211-231.

Chung, Sandra & Timberlake, Alan. 1985. "Tense, Aspect and Mood". *Language Typology and Syntactic Description* ed. by Timothy Shopen, volume 3, 202-248 (chapter 4) Cambridge, U.K.: Cambridge University Press.

Collins, Michael. 1996. "A New Statistical Parser Based on Bigram Lexical Dependencies". *Proceedings of the 34th Annual Meeting of the Association of Computational Linguistics* (*ACL'96*), 184-191. Santa Cruz, Calif.

Filatova, Elena & Eduard Hovy. 2001. "Assigning Time–Stamps to Event–Clauses". *Proceedings of the Workshop on Temporal and Spatial Information Processing at ACL'01*, 445-482. Toulouse, France.

Fiscus, Jon, George Doddington, John Garofolo & Alvin Martin. 1999. "NIST's 1998 Topic Detection and Tracking Evaluation (TDT2)". *Proceedings of the 1999 DARPA Broadcast News Workshop*, 19-24. Herndon, Virginia, U.S.A.

Harman, Donna & Daniel Marcu, eds. 2001. *Proceedings of the Document Understanding Conference* (*DUC-2001*). NIST, New Orleans, U.S.A.

Levin, Beth. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. Chicago, Ill.: University of Chicago Press.

Marsh, Elaine & Dennis Perzanowski. 1997. "MUC-7 Evaluation of IE technology: Overview of Results". *Proceedings of the 7th Message Understanding Conference* (*MUC-7*). Fairfax, Virginia, U.S.A. — `http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/proceedings_index.html` [Source checked in May 2004]

McCoy, Kathleen F. & Michael Strube. 1999. "Taking Time to Structure Discourse: Pronoun Generation Beyond Accessibility". *Proceedings of the 1999 Meeting of the Cognitive Science Society*, 378-383. Vancouver, British Columbia, Canada.

Miller, George, Richard Beckwith, Christiane Fellbaum, Derek Gross & Katherine Miller. 1990. "Introduction to WordNet: An Online Lexical Database". *International Journal of Lexicography* 3:4.235-312.

Pustejovsky, James. 2000. "Events and the Semantics of Opposition". *Events as Grammatical Objects* ed. by C. Tenny & J. Pustejovsky, 445-482. Stanford, Calif.: CSLI Publications.

Siegel, Eric V. & Kathleen McKeown. 2000. "Learning Methods to Combine Linguistic Indicators: Improving Aspectual Classification and Revealing Linguistic Insights". *Computational Linguistics* 26:4.595-628.

Yang, Yiming, Jaime Carbonell, Ralf Brown, Thomas Price, Brian Archibald & Xin Liu. 1999. "Learning Approaches for Detecting and Tracking News Events". *IEEE Intelligent Systems: Special Issue on Applications of Intelligent Information Retrieval* 14:4.32-43.

Semantically Driven Approach for Scenario Recognition in the IE System FRET

SVETLA BOYTCHEVA*, MILENA YANKOVA** & ALBENA STRUPCHANSKA**

* *Sofia University "St. Kl. Ohridski"*
** *Bulgarian Academy of Sciences*

Abstract

This paper reports a research effort in the scenario recognition task in Information Extraction (IE). The presented approach uses partial semantic analysis based on logical form representation of the templates and the processed text. It is implemented in the system FRET[1] (Football Reports Extraction Templates), which processes specific temporally structured texts. A logical inference mechanism is used for filling template forms where only scenario-relevant relations between events are linked in the inference chains. Some aspects of negation and modalities that occur in the texts are also taken into account.

1 Introduction

The most important and difficult sub-task in IEis scenario recognition. There are many systems that address this problem applying different solutions. Roughly, these solutions can be characterised as shallow or deep depending on the processing depth at each of the stages. During the syntactic analysis the processing varies from phrasal chunking to parsing and produces regularised forms (anything from partially filled templates to full logical forms).

The discourse or multi-sentence level processing that follows the syntactic analysis can be also more or less deep. It depends on the usage of declaratively represented world and domain knowledge to help resolving ambiguities of attachments, word sense, quantifiers scope, and coreferences or to support inference-driven templates filling (Gaizauskas & Wilks 1998). One of the last presented systems using domain specific knowledge as semantic net is LaSIE (Gaizauskas et al. 1995, Humphreys et al. 1998). It attempts fragmentary parsing only and falls somewhere in between the deep and the shallow approaches.

Most systems from latest MUC (Message Understanding Conferences) avoid usage of deep processing as it is hard to accomplish many of the required natural language understanding tasks. The great effort needed for building up the domain knowledge is another reason for using shallow processing. Some authors (Grishman 1997) believe that only grammatical relations relevant to the template

[1] This work is partially supported by the European Commission via contract ICA1–2000–70016 "BIS–21 Centre of Excellence".

should be considered. A good example for shallow approach implementation is FASTUS (Appelt et al. 1993, 1995). It provides phrase parser and recognises domain patterns for building raw templates that are normalised by the postprocessor. The authors reported on difficulties in defining all possible patterns for a given event though FASTUS showed second best results on MUC–5.

Our system focuses on template filling— an important and still open question in our view. Many systems (Yangarber et al. 2000) for (semi-) automatic generation of templates based on machine learning approach exist. So we assume that templates are given beforehand and FRET tries only to recognise scenario and to fill the corresponding templates correctly.

In this paper we present semantically driven approachfor scenario pattern matchingin the IE system FRET (Yankova & Boytcheva 2003). Our approach is to provide deep understanding only in "certain scenario-relevant points" by elaborating the inference mechanisms. So we decided to stay between deep and shallow approaches for text processing.

The paper is organised as follows: Section 2 presents an overview of FRET architecture. Section 3 discuses our improvements in translation to logical forms(LFs), especially the coreference resolution and the recognition of negation and modalities, which appear in the chosen domain. Section 4 describes the inference mechanism in FRET. Evaluation results are in Section 5. Section 6 contains the conclusion and sketches some directions for further work.

2 FRET Architecture

The design and the development of the Knowledge Base (KB) and the system architecture are influenced on the football domain specificity. On the one hand the football records paragraph structure (with tickers for each minute) provides rich temporal information that simplifies the choice of text parts to be processed in search of the scenario. On the other hand scenario recognition in a frequently changeable domain is a hard task. Both the domain terminology and statements in football reports are fast changing. There is no certainty in the truth of the stated facts as they could be negated later. Also the specific usage of words, which are treated as terms in the domain, is embarrassing even for human beings.

The system consists of the three main modules: text preprocessor, logical form translator and templates filler. The text preprocessor performs lexical analysis, Name Entity (NE) recognition and part-of-speech tagging of football reports. GATE system (Cunningaham et al. 2002) is integrated in FRET and its modules perform these tasks.

The coreference resolution task is performed by the logical form translator. Its algorithm takes into account the domain characteristics. However, it solves neither the usage of metaphors and nicknames of football players nor the variety of foreign players names with their transcriptions. These problems reflect

on the performance of the NE recognition and respectively on the coreference resolution.

FRET's KB contains two main parts: static and dynamic resource banks. The static resource bank includes lexicon, grammar rules, rules for translation in LFs, templates description and description of the domain events with their relations. All uninstantiated events LFs and relations between them are presented as graph nodes and arcs respectively. Some information concerning the team/coach names, players' list, playing roles, penalties, etc. is not constant so it could not be added to the static part of the KB. It is included into the dynamic resource bank and is automatically collected for each football report during the text processing. Both the logical form translator and the templates filler use the KB.

3 Logical form translation

Usually in NL texts, fragments of partial information about an event are spread over several sentences. These descriptions need to be combined before the scenario recognition. That is why FRET associates the time of the event to each produced LF. Every LF is decomposed into its disjuncts and each of them is marked with the associated time.

A specially developed partial syntactic parser implemented in Sicstus Prolog is used in FRET for logical form translation. All words in LF are represented as predicates, where the predicate symbol is the corresponding base form of the word and has one argument. Specially denoted predicates with a symbol "θ" and an arity 3 are used for representation of the thematic roles (see Example 1). The arguments describe the thematic role and its relations with the corresponding predicates. In the case of proper name, the argument is substituted by the string of the name.

Example 1:

Sentence:

17 mins: Beckham fires the ball into Veron.

Logical form:

```
time(17) & fire(A) & θ(A, agnt, 'Beckham') & ball(D)
& θ(A, obj, D) & θ(A, into, 'Veron').
```

Some aspects of coreference resolution (Dimitrov 2002) are also solved at this stage. Only pronominal and proper names coreference are important for the discussed system. The pronominal one is the most common type of coreference in the chosen domain. Pronominal coreference resolution is restricted according to the domain only in detection of the proper antecedent for the following masculine pronouns: (*i*) personal: *he, him* and (*ii*) possessive: *his.* These anaphora are solved by binding the pronoun to the nearest left position agent in the extended

paragraph, which includes the last sentence from the previous minute, all sentences from the current minute and the first sentence from the next minute. The relative pronoun *who* is solved by binding the pronoun to the nearest left position noun in the current sentence. The proper names coreference is based on the players' lists from the dynamic resource bank.

Another problem that has to be solved during the parsing process is the identification of negations. As described in (Boytcheva et al. 2002) and taking into account the specific domain texts, we distinguish explicit and implicit negations. As total 52% of sentences in our corpus contain negations: 14% explicit negations and 38% implicit ones.

We consider two types of explicit usage of negation:

- Short sentence containing only "No". In this case we mark the LF of previous sentence with marker for negation "NEG";
- Complete sentence, containing "Not/Non/No". In this case the scope of the negation is the succeeding part of the current sentence and thus we mark only its LF with marker for negation "NEG".

In implicit usage of negation (words as "but", "however", "although"...) (Mastop 2001), both LFs of words preceding and succeeding the negation in the sentence (in some cases previous or next sentence) are marked with markers for negation: 'BAHpos' and 'BAHneg' (see Example 2, BAH is an abbreviation for But, Although, However and "7" is an internal identification for the marker).

Example 2:

Sentence:

79 mins: Henry fires at goal, but misses from a tight angle.

Logical forms:

```
time(79) & fire(A) & θ(A,agnt,'Henry') & θ(A,at,B)
& goal(B) & marker('BAHpos',7).
time(79) & miss(A) & θ(A,agnt,'Henry') & θ(A,form,B)
& angle(B) & θ(B,char,C)& tight(C) & marker('BAHneg',7).
```

Another problem that has to be solved is to recognise sentences with modals ("*can*", "*should*", "*may*", "*have to*", ...) that appear in about 10% of the test corpus. In this case we mark their LFs and the LF of the next sentence with marker "MOD", because we expect acceptance or rejection of the current modality in the next sentence (see Example 3).

Example 3:

Sentences:

21 min: Jaap Stam will be next. Surely he has to score. GOAL!!!
The ball reached the back of the net.

Logical forms:

```
time(21) & score(A) & theta(A,agnt,'Jaap Stam')
& theta(A,char,B) & surely(B) & marker('MOD',6).
```

All markers are necessary for further inference as at first we only recognise modalities and possible negations and postpone their interpretation.

4 Inference mechanism

In FRET all templates are described as tables with: (*i*) *obligatory* and (*ii*) *optional* fields that have to be filled in. Both types of fields, taken as a whole, contain the key information presented in the text (Wilks 1997). We state that the scenario is recognised if at least the obligatory fields are filled in, while the optional fields can be left empty.

In FRET we distinguish three types of events related to each scenario that are structured into a directed graph (see Fig. 1) – preliminary stored in the static resource bank :

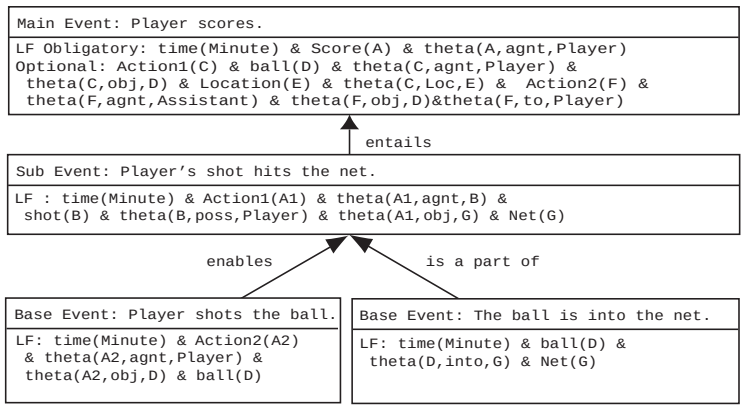

Figure 1: *A part of the graph, stored in the static resource bank*

- *main event:* the template description as LF of obligatory and optional fields and relations between them;
- *base events:* LFs of the most important self-dependent domain events;
- *sub-events:* kinds of base events immediately connected to the main one (there is an arc between the nodes of the main and the sub-events).

The matching algorithm of FRET is based on the relations between events. Each of these relations is represented as an arc with associated weight in the graph. We use four types of relations, defined as follows:

- Event E_2 **invalidates** event E_1, i.e., event E_2 happens after E_1 and annuls it.

- Event E_1 **entails** event E_2, i.e., when E_1 happens E_2 always happens at the same time.
- Event E_1 **enables** event E_2, i.e., event E_1 happens before the beginning of event E_2 and event E_1 is a precondition for E_2.
- Event E_2 **is a part of** event E_1.

The template filling module of FRET performs two main steps: (*i*) matching LFs and (*ii*) template filling . The step of matching LFs is based on the modification of the unification algorithm. We are interested in those LFs, which are produced from the so-called extended paragraph. Thus we process each paragraph separately.

Initially the module tries to match each LF from the extended paragraph to the main event. We call this step direct matching. The direct matching algorithm succeeds when at least one LF is matched to the main event. When there are no markers for negation or modalities in the matched LF, the system proceed with the template filling algorithm. Otherwise, the availability of such markers is an indication that there is no certainty in the truth of the statement matched to the main event. So, some additional steps are necessary for correctly processing the marked LFs and therefore recognising an event. These steps depend on the attached types of markers.

- NEG – the result from the matching algorithm is ignored, so the event is not recognised. In this case we consider that it is better not to recognise some event than to recognise it wrongly.
- BAHpos, BAHneg – we treat LFs with these markers according to the semantic interpretation of contrastive particles (Winter & Rimon 1994). There are two major interpretation types: as negation and as conjunction of independent statements. We are interested only in the contrast interpretation, that is why we have to check whether we really have negation in the marked LFs. The latter are processed as follows: all events related by invalidate relation to the matched event are collected into a set; the LF from the extended paragraph, which is marked with the other BAH marker is juxtaposed to a member of the collected set; if this succeeds, the previous matching is ignored.
- MOD – the event is correctly recognised only when the next LF from the extended paragraph contains explicit confirmation (Sahlqvist 1975) (e.g., Yes; GOAL!!!; Bravo etc.) or the same scenario is recognised in other sentence from the current paragraph.

The direct matching algorithm succeeds when all variables of main event's LFs related to obligatory fields are bound. Synonyms lists from the KB are used for the unification of variable predicate names in LFs.

If the direct matching algorithm fails FRET starts the inference matching algorithm (cf. Figure 2). which is described below in a more formal way.

EP: LFs included in current extended paragraph
M: main event
G: corresponding graph in KB
C: predefined coefficient

$S ::= \{E_i\}_{i=1}^{n}$ where E_i is a sub-event of M

for $i = 1..n$:

$B ::= \{B_j\}$ where B_j is a base event from G and
$\exists\ arc(B_j, E_i, r_j) \in G$

Apply the direct matching algorithm to all possible couples (L_k, B_p) where $L_k \in EP$ and $B_p \in B$. Collect successfully matched couples in set B'.

Calculate $R = \sum_{j=1..|B'|} r_j$ where $arc(B_j, E_i, r_j) \in G$ and $B_j \in B'$

if $R \geq C$ **then** "Unify" & SUCCESS**; break**

if $\neg$ SUCCESS **then** FAIL

Figure 2: *Inference-matching algorithm*

When the inference-matching algorithm succeeds, all possible predicates from the selected sub-event (E_i) are unified by corresponding variables from the predicates in the set of successfully matched base events (B'). Thus the sub-event LF is successfully matched. A similar unification is applied to the set of matched sub- and base-events ($E_i \cup B'$) and the main event (M) in order to fulfil the main event's LF (as shown in Fig. 1).

The templates filling starts only if either the direct or the inference matching algorithm succeeds. Initially the template obligatory fields are filled by the required information. At this stage some additional information from the dynamic resource bank is used too. The completion of all obligatory fields is sufficient for correct scenario recognition. However, the optional fields, for which enough information exist, are also filled in.

Example 4:

53 mins: Beckham shoots the ball across the penalty area to Alan Shearer who heads into the back of the net at the far post.

Figure 3 shows the result obtained by FRET after processing a football match, which contains sentences shown in Example 4 . The GOAL scenario is recognised in the paragraph marked with *"53 mins"*. The direct matching failed but

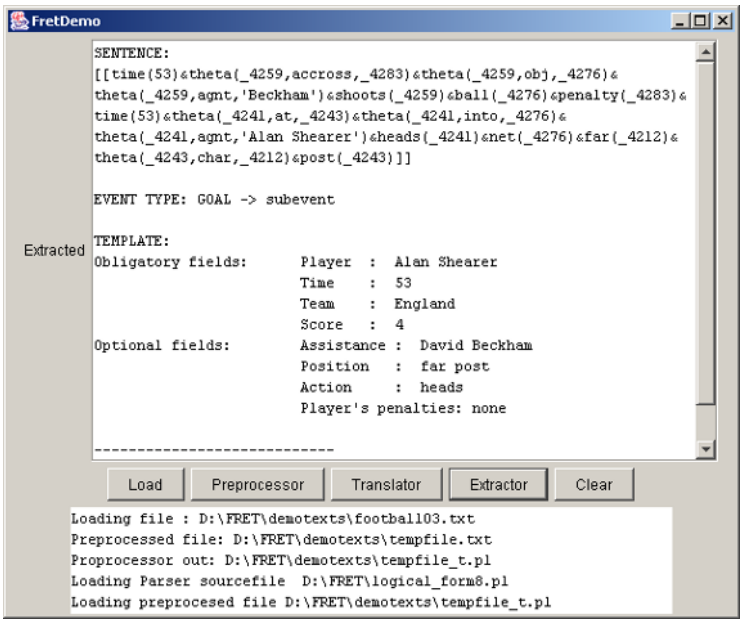

Figure 3: *The result obtained after filling in a template from Example 3*

the inference matching algorithm succeeded in matching one of the sub-events (i.e., *"Player heads into the net"*). Thus the obligatory fields are filled with the corresponding information form the text (i.e., *player name = "Alan Shearer"*) and from the dynamic resource bank (i.e., *team = "England"*). Please note that in this case there is enough information for completing all optional fields but this is an optimistic sample case.

5 Evaluation

FRET is tested only on the scenario "goal" for the texts of 50 reports, which totally contain 148 instances of the event "goal".

The f-measure of the text preprocessor module is $< 96\%$ and depends on GATE performance. The algorithm developed for coreference resolution has f-measure $< 89\%$ (the percentage is high because we are interested only in particular cases of pronominal and proper noun coreference). The f-measure of the parser is $< 91\%$.

The domain specific treatment of negation and modalities improve the f-measure of the FRET system. So the scenario templates are filled in with pre-

cision: 86%, recall: 61 % and f-measure: 71.37 %. The direct matching works in 12% and the inference is applied in 88% of the cases.

6 Conclusions and further work

Scenario recognition is important and difficult task for IE. So in this paper we make an attempt to find an easy and effective way for scenario recognition that may facilitate semantic processing of large text collections.

With the usage of inference we can find either similar ways of expressing different scenarios or partial information about the same event spread over several sentences. Thus we believe that the inference is an integral part of finding facts in texts. In order to make effective inference it is necessary to represent sentences into LFs and to have suitable representation of the domain knowledge. We have to emphasize on the major role of the specially tailored structure of the events and relations between them as a graph. The choice of simple relations between events makes the inference mechanism in the graph structure easier. However, not all the information provided in the text is needed for simple template filling. So our approach uses shallow parsing and partial semantic analysis.

The innovative aspects in FRET at this stage of development are:

- attempts for domain-specific treatment of implicit negation and modalities, as well as
- elaborated inference mechanism that provides relatively deep NL understanding but only in "certain points". Please note that the inference is simple and effective due to the consideration of only scenario relevant relations between events.

Our plans for future development of the system include making a deep investigation of the domain and completing the events graph. This will make our evaluation more precise and probably will improve the results reported above. We also plan to test the system behaviour on other domains with a specific temporal structure.

REFERENCES

Appelt, Douglas E., Jerry R. Hobbs, John Bear, David Israel, Megumi Kameyama, Andrew Kehler, David Martin, Karen Myers & Mabry Tyson. 1993. "SRI: Description of the JV-FASTUS System Used for MUC-5". *Proceedings of the 5th Message Understanding Conference* (*MUC-5*), 221-235. Tokyo, Japan.

Appelt, Douglas E., Jerry R. Hobbs, John Bear, David Israel, Megumi Kameyama & Mabry Tyson. 1995. "SRI International FASTUS System MUC-6 Test Results and Analysis". *Proceedings of the 6th Message Understanding Conference* (*MUC-6*), 237-248. Columbia, Maryland: Morgan Kaufmann.

Boytcheva, Svetla, Albena Strupchanska & Galia Angelova. 2002. "Processing Negation in NL Interfaces to Knowledge Bases". *Proceedings of the International Conference on Conceptual Structures* (*ICCS'02*) (= *Lecture Notes in Artificial Intelligence*, 2393), 137-150. Borovets, Bulgaria.

Cunningaham, Hamish, Diana Mayard, Kalina Bontcheva, Valentin Tablan, Cristian Ursu & Marin Dimitrov. 2002. "The GATE User Guide" — `http://gate.ac.uk/` [Source checked in May 2004].

Dimitrov, Marin. 2002. *A light-weight Approach to Coreference Resolution for Named Entities in Text*. MSc thesis, Sofia University.

Gaizauskas, Robert, Takahiro Wakao, Kevin Humphreys, Hamish Cunningham & Yorick Wilks. 1995. "University of Sheffield: Description of the LaSIE System as Used for MUC-6". *Proceedings of the 6th Message Understanding Conference* (*MUC-6*), 207-220. Columbia, Maryland: Morgan Kaufmann.

Gaizauskas, Robert & Yorick Wilks. 1998. "IE: Beyond Document Retrieval". *Journal of Documentation* 54:1.70-105.

Grishman, Ralph. 1997. "Information Extraction: Techniques and Challenges". *International Summer School on Information Extraction* (*SCIE'97*) *Frascati, Italy* (= *Lecture Notes in Computer Science*, 1299), 10-27. Berlin: Springer.

Humphreys, Kevin, Robert Gaizauskas, Saliha Azzam, Chris Huyck, Brian Mitchell, Hamish Cunningham & Yorick Wilks. 1998. " Univ. of Sheffield: Description of the LaSIE-II System as Used for MUC-7". *Proceedings of the 7th Message Understanding Conference* (*MUC-7*), 84-89. Fairfax, Virginia.

Mastop, Rosja. 2001. "Formalizing the Contrastive Particle". — `www.geocities.com/rosjamastop/rosja_mastop.pdf` [Source checked in May 2004].

Sahlqvist, H. 1975. "Completeness and Correspondence in First and Second Order Semantics for Modal Logic". *Proceedings of the Third Scandanavian Logic Symposium* ed. by S. Kanger, 110-143. Amsterdam: North Holland.

Wilks, Yorick. 1997. "Information Extraction as a Core Language Technology". *International Summer School on Information Extraction* (*SCIE'97*) *Frascati, Italy* (= *Lecture Notes in Computer Science*, 1299), 1-9. Berlin: Springer.

Winter, Yoad & Mori Rimon. 1994. "Contrast and Implication in Natural Language". *Journal of Semantics* 11:365-406.

Yangarber, Roman, Ralph Grishman, Pasi Tapanainen & Silja Huttunen. 2000. "Unsupervised Discovery of Scenario-level Patterns for Information Extraction". *Proceedings of the 6th Applied Natural Language Processing* (*ANLP-NAACL 2000*), 282-289. Seattle, Washington, U.S.A.

Yankova, Milena & Svetla Boytcheva. 2003. "Focusing on Scenario Recognition in Information Extraction". *Proceedings of the European Chapter of the Association for Computational Linguistics* (*EACL'03*), 41-48. Budapest, Hungary.

A Framework for Named Entity Recognition in the Open Domain

RICHARD J. EVANS
University of Wolverhampton

Abstract

In this paper, a system for Named Entity Recognition in the Open domain (NERO) is described. It is concerned with recognition of various types of entity, types that will be appropriate for Information Extraction in any scenario context. The recognition task is performed by identifying normally capitalised phrases in a document and then submitting queries to a search engine to find potential hypernyms of the capitalised sequences. These hypernyms are then clustered to derive a typology of named entities for the document. The hypernyms of the normally capitalised phrases are used to classify them with respect to this typology. The method is tested on a small corpus and its classifications are evaluated. Finally, conclusions are drawn and future work considered.

1 Introduction

Information Extraction (IE) is defined as the automatic identification of selected types of entities, relations, or events in free text (Grishman 2003). Significant multi-site evaluations of IE have been carried out in the Message Understanding Competitions (e.g., MUC-7 (Chinchor 1998)). In the context of IE, the events of interest described in documents are encoded using templates. An IE system attempts to assign the participants of an event to functional *slots* in the template. The templates used in the MUC-7 IE task have slots for PERSON, ORGANIZATION, ARTIFACT, and LOCATION elements. The goal of named entity recognition (NER) is to identify these elements in texts automatically.

MUC-7 relates to one IE scenario but many more types of entity should be recognised for effective IE in different domains. To illustrate, in the domains of medicine or e-retail, systems will need to identify pharmaceutical names or product names. The set of required name types varies from case to case. The NER approaches developed for MUC-7 are able to recognise a small set of named entity (NE) types with considerable accuracy. However, in most cases, the classification models used rely on specific domain knowledge. They cannot easily be extended to recognise the pertinent entities occurring in documents from other domains.

In the current state of the art, IE systems must be re-implemented for new scenarios. In this paper, the goal is to automatically identify the entities that are likely to be of interest in any scenario context, with no knowledge *a priori*. A

system called NERO[1] is described that embodies a framework for NER in the open domain.

The paper is structured as follows. Section 2 describes the methods by which NERO addresses its goals. In Section 3, the small corpus used to test the system is described and in Section 4 the resulting evaluation is reported. In Section 5, related work is reviewed and in Section 6 conclusions are drawn and directions for future research considered.

2 The method for named entity recognition in the open domain

The process of open-domain NER is tackled in three stages. Firstly, a document-specific typology for NEs is derived automatically (Section 2.1). Secondly, NEs are identified (Section 2.2). Thirdly, NEs are classified in line with the derived typology (Section 2.3).

2.1 *Typology derivation*

The typology is obtained by collecting the hypernyms of capitalised phrases, clustering the hypernyms, and labelling those clusters. The method for identification of hyponyms described in Hearst (1992) was applied in order to identify potential hypernyms of sequences of capitalised words appearing in the document. Here, sequences include single words. In the first sentence of the abstract of this paper, the sequences of capitalised words are {*In*, *Named Entity Recognition*, *Open*, and NERO}. Numerous patterns were used to produce queries that were submitted to the google[2] search engine. The summaries returned by *google* were then used to derive the hypernyms. Following Hearst (1992), when X is a capitalised sequence, the query `such as X`, was submitted to *google*. The FDG parser (Tapanainen & Järvinen 1997) was used to find the part of speech and lemma of words in the returned summaries. The lemma of the immediately preceding noun was chosen as the hypernym of X. Three other patterns (Figure 1) were included in an effort to improve coverage. In these patterns, Y is a noun phrase whose head is a potential hypernym of X. When running NERO, queries were submitted for each capitalised sequence and all of its substrings. The first 1000 results from *google* were processed in each case.

In the documents used as the test corpus for this work, the four patterns shown in Figure 1 appear very rarely. In fact, for these documents only 1.19% of NEs appear in those patterns. This contrasts with 96.46% when the Internet is taken as the source of potential hypernyms. This observation justifies exploitation of the Internet for this task.

[1] NERO stands for Named Entity Recognition in the Open domain.
[2] `www.google.com`

Y such as X
Y like X
X or other Y
X and other Y

Figure 1: *Hypernymy patterns*

Having obtained sets of potential hypernyms for all sequences of capitalised words in the input text, the system clusters the global set of hypernyms in an attempt to find the general types of NE that appear in that document. NEs will be classified on the basis of the resultant typology. A hard, bottom-up, hierarchical clustering algorithm was used (Manning & Schütze 1999:500-503). It is presented in Figure 2.

- Given:
  - a set $H := \{\chi_1, ..., \chi_n\}$ of hypernyms
  - a group-average similarity function sim, based on taxonomic similarity.
  - $\Upsilon = 1$
- **for** $i := 1$ **to** n **do**
  $h_i := \{\chi_i\}$ **end**
  $C := \{h_1, ...h_n\}$
  $j := n + 1$
- **while** $\Upsilon > \tau$
  (1) $\Upsilon := max_{(h_u,h_v)\in C\times C} sim(h_u, h_v)$
  (2) $(h_{n_1}, h_{n_2}) := argmax_{(h_u,h_v)\in C\times C} sim(h_u, h_v)$
  (3) $h_j := h_{n_1} \cup h_{n_2}$
  (4) $C := C \setminus \{h_{n_1}, h_{n_2}\} \cup \{h_j\}$
  (5) $j := j + 1$

Figure 2: *The algorithm used to cluster hypernyms*

This type of unsupervised learning is suitable for the current task in which no information on the desired set of derived types is available *a priori*. The similarity function used in the clustering algorithm is a group average method that assesses taxonomic similarity between hypernyms with respect to WordNet (Fellbaum 1998). Taxonomic similarity was computed using *Learning Accuracy* (Hahn & Schnattinger 1998). The clustering process is halted when the similarity between the two most similar clusters drops below some threshold. Empirical observation indicated that a threshold of 0.5 was suitable. The stopping condition is set to prevent hypernyms indicative of distinct types from being merged.

The WordNet package (Fellbaum 1998) was used in order to label the resultant clusters. For all senses of all words in a cluster, increasingly general hypernyms were listed. These lists were compared within the cluster and the most specific hypernym common to all words[3] was used to label the cluster as a whole. We defined a measure, *height*, for each label that is the mean number of nodes between the common hypernym and the senses in the cluster. This measure is used in the classification of named entities (Section 2.3). The set of labels associated with derived clusters forms the typology that will be used as the basis for classification of NEs.

2.2 *Identification of named entities (capitalised word normalisation)*

Capitalisation is one signal that can distinguish NEs from other phrases in texts but it is also used in the general layout and structuring of documents. It is thus necessary to disambiguate capitalisation to determine whether a given word is normally capitalised in all contexts, or whether capitalisation of the word is context dependent. This disambiguation is referred to as *normalisation* (Mikheev 2000).

NERO performs normalisation with a memory based learning method (TiMBL, described in Daelemans et al. (2001)). Each capitalised word in the training data is associated with a vector of feature values and a binary classification (`NORMALLY_CAPITALISED` or `NOT_NORMALLY_CAPITALISED`). Features appearing in the vectors include:

(1) positional information,
(2) the proportion of times the word is capitalised in the document,
(3) the proportion of times the word is sentence initial in the document and in the BNC (Burnard 1995),
(4) whether the instance appears in a gazetteer of person names or, following Mikheev (2000), in a list of the top 100 most frequent sentence initial words in the BNC,
(5) the part of speech of the word and the surrounding words,
(6) agreement of the word's grammatical number with the following verb *to be* or *to have*.

The training data contains 3168 instances. The method was evaluated using ten-fold cross validation. It obtained an overall precision of 98.63% and recall of 98.51% for `NORMALLY_CAPITALISED` instances and 100% precision and 98.31% recall for `NOT_NORMALLY_CAPITALISED` instances.

[3] Not necessarily all senses.

2.3 *Classification of named entities*

Classification of NEs exploits the derived typology. T is a type that subsumes hypernyms $\{t_1, ..., t_m\}$ and ϕ is a coefficient inversely proportional to the *height* of T. Word w matches, or is a substring of, a capitalised sequence C that has hypernyms $\{h_1, ..., h_n\}$ and each hypernym has a frequency fh_i. The likelihood that w should be classified as T is given by:

$$\sum_{j=1}^{m} \sum_{i=1}^{n} \phi . g(h_i, t_j) . fh_i$$

The function $g(h_i, t_j)$ maps to 1 when h_i is identical to t_j, and maps to 0 otherwise. Having computed the likelihood for all types, w is classified as belonging to the one for which this measure is greatest.

3 The test corpus

The creation of evaluation data is very difficult for the non-expert due to the "open" nature of the NE typology. For the pilot study presented in this paper, just nine documents were hand annotated and an assessment of NERO's performance was made against this human annotated data.

DOC	#WORDS	#NES	GENRE
a01	1944	258	Legal
j53	1980	2	Psych.
j59	1959	55	Art
k09	2005	92	Lit.
k12	2020	129	Lit.
k22	2024	137	Lit.
k25	1985	29	Lit.
n05	2051	75	Lit.
win	2884	274	Tech.
TOT	18852	1051	_

Table 1: *Characteristics of the documents used to test* NERO

Of the nine texts, eight were from the SEMCOR corpus (Landes et al. 1998) and one was a technical document, *win*. One point to be made about the documents taken from SEMCOR is that these are extracts from larger texts, and are thus incomplete. As will be noted in Section 4 this has some unfortunate consequences.

4 Evaluation

NERO is evaluated with respect to its ability to identify NEs, to derive an appropriate typology of NEs for a given document, and to classify NEs in line with the typology.

4.1 *Evaluating normalisation*

The method for capitalised word normalisation (Section 2.2) was applied to the documents used to test NERO. Overall, NERO was able to identify normally capitalised words with a precision of 97.48% and recall of 88.39%. For words that are not normally capitalised, the figures were 100% and 97.11% respectively. The difference in performance when processing the nine test documents and that obtained in ten-fold cross validation based on the training data is partially explained by the fact that, as noted in Section 3, the documents from SEMCOR (Landes et al. 1998) are incomplete. Many PERSON NEs are referred to using only a surname or nickname in these files. The full name may have been introduced earlier in the text, but this evidence is missing from the extract available in SEMCOR. This affected NERO's performance as it rendered the gazetteers used in Feature 4 (Section 2.2) redundant in many cases. While most systems for capitalised word normalisation are able to correctly classify nicknames and surnames when they appear, these successful classifications are usually facilitated by the appearance of the full names elsewhere in the document (Mikheev 2000).

4.2 *Evaluating typology derivation*

Evaluation of the typology derivation task is problematic because manual annotation may require a high level of expertise within a given domain in order to classify some types of entity and non-experts will tend to label these NEs with general types. Hiring experts will be an expensive undertaking within the context of annotation in the open-domain. With respect to the NE types found in the test files, several match those that are used in MUC-7 but there are additional types used here. NAT_LAN covers NEs that refer to nationalities or languages. The titles of creative works such as paintings or books are marked CTVE_TTL. The names of menu items or buttons in computer software are marked OPT_BUT.

An assessment was made of system performance in the derivation of a typology. Precision and recall was computed for the set of clusters that constitute types used to label instances in the test files. We define precision as the ratio of the number of machine derived clusters that correlate well with a human derived type, to the total number of machine derived clusters. Recall is defined as the ratio of the number of machine derived clusters that correlate well with a human derived type to the total number of human derived types annotated in the key file. When calculating these figures, it was noted that in 40.91% of cases, the machine derived clusters correlate only partially with human derived types. In these cases, the machine derived cluster was counted as a good match if more than half of the senses in the cluster were felt to be indicative of the human derived type. Otherwise, it was not counted. The precision of the clustering method is poor, at 46.97%. Recall is mediocre, at 67.39%. The performance of the clustering method will limit that of the NE classification task.

Inspection of the degree of correlation between machine derived clusters and human derived types revealed a number of problems. In text *a01*, one cluster was too general, incorporating hypernyms that distinguish two imporant types, ORG and LOC. The problem of word sense ambuiguity was highlighted in the clusters derived for document *k09*. Here, one hypernym *character*, was merged with a cluster of words that share its graphic, rather than human, sense. It would be necessary to perform word sense disambiguation (WSD) to solve this problem.

It was also clear that clustering is influenced by the hypernym collection process. A qualitative assessment of the clusters produced by NERO showed that many NEs are classified by clusters of ambiguous hypernyms. To illustrate, the hypernyms automatically obtained for TIME expressions such as *Monday*, *Wednesday*, *May*, etc. tended to be more indicative of PERSON NEs than TIME ones. Disambiguation of such hypernyms, by rule-based or statistical methods, will be necessary in order to circumvent this problem. Another common source of error (poor recall) in the automatic collection of hypernyms was caused by the use of possessive forms of NEs in the test documents. An unforeseen problem during development of the system, it is expected that this issue can be solved by the simple application of stemming rules during the hypernym collection process.

The simple formulation of hypernym collection patterns causes valuable context to be ignored to a large extent. For instance, where X is the lemma of a noun that is a potential hypernym of the NE Y, the pattern X ... *like* ... Y fails to draw a distinction between useful patterns such as ***people*** *like* ***Peter*** in the summaries returned by *google*, and less useful ones like ***John*** *doesn't like* ***Peter***. In other examples, *google*'s replacement of textual material in the returned summaries by '...' is a source of error that leads to the derivation of unusual sets of hypernyms for NEs. As an example, the name *God* was classified by the cluster {*word*, *book*} when NERO is set with high values for the similarity threshold. This issue should be resolved by filtering certain strings that are returned by the search engine. These filters should be encoded in terms of both lexical items and punctuation marks.

4.3 *Evaluating NE classification*

Table 2 summarises the performance of the system in classifying the NEs in a text. The column NORM indicates the accuracy with which NERO is able to identify the NEs in a document. #NEsID*d* gives the exact number of capitalised words identified by NERO. It can be compared with the column #NE*s* in Table 1 which shows the actual number of words used in NE references in the document. The remaining columns in Table 2 show the percentage of instances classified correctly (CORR) or incorrectly (INCORR). In many cases, the queries submitted to *google* are unable to indicate any potential hypernyms for a capitalised se-

DOC	NORM (%)	#NESIDD	CORR (%)	INCORR (%)	UNCLASS'D (%)
a01	91.13	227	33.48	66.52	42.29
j53	96.92	3	33.33	66.67	0
j59	89.91	42	52.38	47.62	4.76
k09	93.88	46	45.65	54.35	10.87
k12	87.29	101	16.83	83.17	38.61
k22	92.47	90	20.00	80.00	63.33
k25	88.51	22	50.00	50.00	31.81
n05	98.29	37	70.27	29.73	10.81
win	92.48	253	70.36	29.64	2.37
TOT	92.25	821	45.07	54.93	26.31

Table 2: *Performance of NERO on named entity classification tasks*

quence. When this happens, NERO uses any hypernyms that have been found for substrings of the sequence. When no potential hypernyms are available for any substrings, then the instance remains unclassified. The percentage of cases for which this occurred in a document appears in the column UNCLASS'D. Note that these cases are included in the figures under INCORR.

5 Related work

Research activity in IE and NER since the mid-90s has left a large literary footprint. In the first instance, readers are directed to the proceedings of MUC-7 (Chinchor 1998) for a description and evaluation of competing NER systems for English. Similar competitions have been held with respect to Japanese in the IREX (Sekine 1999) conferences.

The continuing influence of the MUC competitions on the fields of IE and NER is significant. Papers such as Zhou & Su (2002), and the shared task at the CONLL Workshop at ACL 2003 address the classification of NEs on the basis of the typology originally used in MUC-7.

The development of an extended NE hierarchy, including more than 150 NE types, is described in Sekine et al. (2002). The researchers involved in this work have also developed a classification system for the extended set of NEs. However, the system is based on manually proposed rules specific to each type of entity and is not strictly robust in the open domain.

As mentioned in Section 2.2, the domain-independent task of text normalization has been addressed before, and with greater accuracy than the system reported in this paper, by Mikheev (2000). He used a *maximum Entropy* classification model which also incorporated information about abbreviations and the occurrence of capitalised sequences in the document.

6 Conclusion

This paper has presented a framework for NER in the open domain, and has described an implemented system, NERO, that embodies this framework (Section 2). It includes automatic components for derivation of a typology of NEs (Section 2.1), for normalisation of capitalised words (Section 2.2), and for classification of NEs in a given document (Section 2.3). The paper has described these components and conducted a small-scale evaluation (Section 4). Related work has also been reviewed (Section 5).

The unsupervised nature of the approach, its independence from annotated training data, and the fact that it has addressed the open-domain are all strengths of the system. The fact that, to a certain extent, NERO is able to derive appropriate typologies of NEs for a variety of documents is also a favourable aspect of its performance. Unfortunately, these strengths are over-balanced by numerous weaknesses. Firstly, the patterns used to collect suitable hypernyms for capitalised sequences are vulnerable to data sparseness. In many cases, no suitable hypernyms were identified by NERO. This problem can be addressed by formulating additional patterns, and by using alternative IR technology that recognises punctuation symbols in queries. However, despite the rapid growth of the Internet, this problem of data sparseness is unlikely to be eliminated. Alternative approaches will be required in order to obtain the semantic type of entities for which the current hypernym collection method fails.

As noted in Section 4, word sense ambiguity is a major problem in the hypernym collection and clustering processes that NERO performs. In future work, it will be interesting to assess the feasibility of using a method for WSD in the hypernym collection and clustering phases. Soft clustering algorithms, in which elements may belong to more than one cluster, are another potential solution to the problem of word sense ambiguity.

It will be useful to experiment with different classification methods for the identified NEs. The weighting on different types can be adjusted not only with respect to the *height* of the label, but also with respect to the size of the cluster, or information from WSD.

The evaluation reported in this paper has been insufficient. In future work it may be useful to apply NERO to the MUC-7 data in order to assess the effectiveness of the typologies derived from those documents. An alternative approach will be to incorporate NERO into different IE systems and obtain extrinsic evaluation results.

The overall value of the framework proposed in this paper remains an open question. The current performance by NERO of the clustering, normalisation, and classification tasks does leave much scope for improvement. These processes must be improved and the system re-evaluated before there are sufficient grounds for accepting or rejecting the approach suggested here.

REFERENCES

Burnard, Lou. 1995. *Users Reference Guide British National Corpus Version 1.0*. Oxford: Oxford University Computing Services.

Chinchor, Nancy A. 1998. *Message Understanding Conference Proceedings*. Maryland: National Institute of Standards and Technology.

Daelemans, Walter, Jakub Zavrel, Ko van der Sloot & Antal van den Bosch. 2001. T*i*MBL*: Tilburg Memory Based Learner, Version 4.0, Reference Guide*. Tilburg, The Netherlands: Tilburg University.

Fellbaum, Christiane. 1998. *Wordnet: An Electronic Lexical Database*. London: MIT Press.

Grishman, Ralph. 2003. "Information Extraction". *The Oxford Handbook of Computational Linguistics* ed. by Ruslan Mitkov, 545-559. Oxford: Oxford University Press.

Hahn, Udo & Klemens Schnattinger. 1998. "Towards Text Knowledge Engineering". *The fifteenth National Conference on Artificial Intelligence (*AAAI/ IAAI*), 26-30 July 1998, Madison, Wisconsin*, 524-531. Menlo Park, Calif.: American Association for Artificial Intelligence.

Hearst, Marti. 1992. "Automatic Acquisition of Hyponyms from Large Text Corpora". *Proceedings of the 14th International Conference on Computational Linguistics* (*COLING-92*), *Nantes, France, 23-28 August 1992*, 539-545. San Francisco: Morgan Kaufmann.

Landes, Shari, Claudia Leacock & Randee I. Tengi. 1998. "Building Semantic Concordances". *WordNet: An Electronic Lexical Database* ed. by C. Fellbaum, 199-216. London: MIT Press.

Manning, Christopher D. & Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. London: MIT Press.

Mikheev, Andrei. 2000. "Document Centered Approach to Text Normalization". *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (*SIGIR-2000*), *Athens, Greece, 24-28 July 2000*, 136-143. New York: Association for Computing Machinery.

Sekine, Satoshi. 1999. *Information Retrieval and Extraction Excercise*. Tokyo: CSL Sony.

Sekine, Satoshi, Kiyoshi Sudo & Chikashi Nobata. 2002. "Extended Named Entity Hierarchy". *Proceedings of the Third International Conference on Language Resources and Evaluation* (*LREC-2002*), *Las Palmas de Gran Canaria, Spain, 29-31 May 2002*, 1818-1824. Paris: ELRA - European Language Resources Association.

Tapanainen, Pasi & Timo Järvinen. 1997. "A Non-projective Dependency Parser". *5th Conference of Applied Natural Language Processing* (*ANLP'97*), 64-71. Washington, D.C.

Zhou, GuoDong and Jian Su. 2002. "Named Entity Recognition Using an HMM-Based Chunk Tagger". *40th Annual Meeting of the Association for Computational Linguistics* (*ACL'02*), 473-480. Philadelphia, Pennsylvania.

Latent Semantic Analysis and the Construction of Coherent Extracts

TRISTAN MILLER

German Research Center for Artificial Intelligence

Abstract

We describe a language-neutral automatic summarization system which aims to produce coherent extracts. It builds an initial extract composed solely of topic sentences, and then recursively fills in the topical lacunæ by providing linking material between semantically dissimilar sentences. While experiments with human judges did not prove a statistically significant increase in textual coherence with the use of a latent semantic analysis module, we found a strong positive correlation between coherence and overall summary quality.

1 Introduction

A major problem with automatically-produced summaries in general, and extracts in particular, is that the output text often lacks fluency and organization. Sentences often leap incoherently from topic to topic, confusing the reader and hampering his ability to identify information of interest. Interest in producing textually coherent summaries has consequently increased in recent years, leading to a wide variety of approaches. Unfortunately, many of these techniques are tied to a particular language or require resources such as a list of discourse keywords and a manually marked-up corpus; others are constrained in the type of summary they can generate (e.g., general-purpose vs. query-focussed).

We present a new, recursive method for automatic text summarization which aims to preserve both the topic coverage and the coherence of the source document, yet has minimal reliance on language-specific NLP tools. Only word- and sentence-boundary detection routines are required. The system produces general-purpose extracts of single documents, though it should not be difficult to adapt the technique to query-focussed summarization, and may also be of use in improving the coherence of multi-document summaries.

Our system fits within the general category of IR-based systems, but rather than comparing text with the standard vector-space model, we employ latent semantic analysis (LSA) (Deerwester et al. 1990), a technique originally developed to circumvent the problems of synonymy and polysemy in IR. LSA extends the traditional vector-space document model with 'singular value decomposition', a process by which the term–sentence co-occurrence matrix representing the source document is factored into three smaller matrices of a particular form.

One such matrix is a diagonal matrix of 'singular values'; when one or more of the smallest singular values are deleted and the three matrices multiplied together, the product is a least-squares best fit to the original matrix. The apparent result of this smearing of values is that the approximated matrix has captured the latent transitivity relations among terms, allowing for identification of semantically similar sentences which share few or no common terms withal. We believe that the deep semantic relations discovered by LSA may assist in the identification and correction of abrupt topic shifts between sentences.

2 Algorithm

The input to our summarizer is a text document converted into a list of m tokenized sentences. The list (indexed from 1 to m) is then segmentedinto linearly discrete topics. This can be done manually if the original document is structured (e.g., a book with chapters), or a linear text segmentation algorithm can be used. The output of this step is a list of sentence indices $\langle t_1, \ldots, t_{n+1} \rangle$, where, for the ith of the n topics, t_i is the index of the first sentence of the topic segment and $t_{i+1} - 1$ is the index of the last sentence of the topic segment. We stipulate that there are no sentences which do not belong to a topic segment, so for all t_i, we have $t_i < t_{i+1}$, and

$$t_i = \begin{cases} 1 & \text{if } i = 1; \\ m + 1 & \text{if } i = n + 1; \\ \text{the index of the first sentence of the } i\text{th topic} & \text{otherwise.} \end{cases}$$

As mentioned previously, we use LSA to measure semantic similarity, so before we can begin constructing the extract, we need to construct a reduced-dimensionality term–sentence co-occurrence matrix. Once this is done, a preliminary extract is produced by choosing a representative 'topic sentence' from each segment — that is, that sentence which has the highest semantic similarity to all other sentences in its topic segment. These topic sentences correspond to a list of sentence indices $\langle r_1, \ldots, r_n \rangle$ such that

$$r_i = \operatorname*{arg\,max}_{t_i \le j < t_{i+1}} \sum_{k=t_i}^{t_{i+1}-1} \operatorname{sim}(j, k),$$

where $\operatorname{sim}(x, y) \in [-1, 1]$ is the LSA cosine similarity score for the sentences with indices x and y. In order to preserve important information which may be found at the beginning of the document, and also to account for the possibility that the document contains only one topic segment, we always consider the first sentence of the document to be a topic sentence — i.e., $r_0 = 1$ — and include it

in our initial extract.[1] Let us refer to this initial extract as $E_0 = \langle e_{0,1}, \ldots, e_{0,n+1} \rangle$ where $e_{0,i} = r_{i-1}$.

As we might imagine, this basic extract will have very poor coherence, since every sentence addresses a completely different topic. However, we can improve its coherence by selecting from the set $\langle 1, \ldots, m \rangle \setminus E_0$ a number of indices for 'glue' sentences between adjacent pairs of sentences represented in E_0. An appropriate glue sentence between two others is one which occurs between them in the source document, and is semantically similar to both. Thus we seek indices $G_1 = \langle g_{1,1}, \ldots, g_{1,n} \rangle$ such that

$$g_{1,i} = \underset{e_{0,i} < j < e_{0,i+1}}{\arg\max} \; f\left(\mathrm{sim}'\left(j, e_{0,i}\right), \mathrm{sim}'\left(j, e_{0,i+1}\right)\right),$$

where

$$f(x, y) = xy \cdot (1 - |x - y|)$$

and

$$\mathrm{sim}'(x, y) = \begin{cases} 0 & \text{if } \mathrm{sim}(x, y) > \alpha \text{ or } \mathrm{sim}(x, y) < 0; \\ \mathrm{sim}(x, y) & \text{otherwise.} \end{cases}$$

for $\alpha \in [0, 1]$. The purpose of $f()$ is to reward glue sentences which are similar to their boundary sentences, but to penalize if the similarity is too biased in favour of only one of the boundaries. The revised similarity measure $\mathrm{sim}'()$ ensures that we do not select a glue sentence which is nearly equivalent to any one boundary — such a sentence is redundant. (Of course, useful values of α will be 1 or close thereto.)

Once we have G_1, we construct a revised extract $E_1 = \langle e_{1,1}, \ldots, e_{1,2n+1} \rangle = \langle E_0 \cup G_1 \rangle$.[2] More generally, however, we can repeat the gluing process recursively, using E_i to generate G_{i+1}, and hence E_{i+1}. The question that arises, then, is when to stop. Clearly there will come a point at which some $e_{i,j} = e_{i,j+1} - 1$, thus precluding the possibility of finding any further glue sentences between them. We may also encounter the case where for all k between $e_{i,j}$ and $e_{i,j+1}$, $f\left(\mathrm{sim}'\left(k, e_{i,j}\right), \mathrm{sim}'\left(k, e_{i,j+1}\right)\right)$ is so low that the extract's coherence would not be significantly improved by the addition of an intermediary sentence. Or, we may find that the sentences with indices $e_{i,j}$ and $e_{i,j+1}$ are themselves so similar that no glue is necessary. Finally, it is possible that the user wishes to constrain the size of the extract to a certain number of sentences, or to a fixed percentage of the original document's length. The first of these stopping conditions is straightforward to account for; the next two can be easily handled by introducing

[1] In practice, it may be the case that $r_1 = 1$, in which case inclusion of r_0 is not necessary. We assume, without loss of generality, that $r_1 \neq 1$.

[2] For notational convenience, we take it as understood that the sentence indices in the extracts E_i are sorted in ascending order — that is, $e_{i,j} < e_{i,j+1}$ for $1 \leq j < |E_i|$.

Algorithm 1: $\mathrm{glue}()$

input : initial extract E, maximum extract length ℓ
output : largest coherent extract of length $\leq \ell$
precondition: $|E| < \ell$
assumption : Lists are kept sorted in ascending order. Where list elements are coordinate pairs, the sorting key is the first coordinate.

$G \leftarrow \langle\rangle$;
for $i \leftarrow 1$ ***to*** $|E| - 1$ **do**
  $s \leftarrow \mathrm{sim}(E[i], E[i+1])$;
  if $E[i] = E[i+1] - 1$ ***or*** $s > \beta$ **then continue**;
  $g \leftarrow \underset{E[i]<j<E[i+1]}{\arg\max} f(\mathrm{sim}'(j, E[i]), \mathrm{sim}'(j, E[i+1]))$;
  if $f(\mathrm{sim}'(g, E[i]), \mathrm{sim}'(g, E[i+1])) \geq \gamma$ **then** $G \leftarrow G \cup \langle (s, g) \rangle$;
end
if $|G| = 0$ **then**
  return E;
else if $|E| + |G| \geq \ell$ **then**
  return $E \cup \left\langle x \;\middle|\; (y, x) \in \bigcup_{i=|E|+|G|-\ell+1}^{|G|} G[i] \right\rangle$;
else
  return $\mathrm{glue}(E \cup \langle x \mid (y, x) \in G \rangle, \ell)$;
end

two fixed thresholds β and γ: when the similarity between adjacent sentences from E_i exceeds β, or when the value of $f()$ falls below γ, no glue sentence is suggested for the pair in question.

The case of maximum summary length is a bit trickier. If we are not concerned about undershooting the target length ℓ, then we can simply halt the algorithm once $|E_i| \geq \ell$, and then take E_{i-1} (or E_i, if $|E_i| = \ell$) as the final extract. Most real-world applications, however, demand that we maximize the extract size. Given E_{i-1} of length $\ell - p$, the optimal extract E of length ℓ is the one which glues together the p largest gaps in E_{i-1}.

A version of the gluing algorithm which takes into account all four stopping conditions is shown in Algorithm 1.

2.1 *Complexity analysis*

Given an initial extract of length n, the first recursion of Algorithm 1 will add at most $n - 1$ sentences to the extract, yielding a new extract of length $2n - 1$. In

general, at most $2^{i-1}n$ sentences will be added on the ith recursion, bringing the extract length to $2^i n - 1$ sentences. Therefore, to achieve an extract of length $\ell > n$, the algorithm needs to recurse at least

$$\left\lceil \log_2 \frac{\ell + 1}{n} \right\rceil$$

times. The worst case occurs when $n = 2$ and the algorithm always selects a glue sentence which is adjacent to one of the boundary sentences (with indices e_1 and e_2). In this case, the algorithm must recurse $\min(\ell, e_2 - e_1)$ times, which is limited by the source document length, m.

On each recursion i of the algorithm, the main loop considers at most $m - (2^i n - 1)$ candidate glue sentences, comparing each one with two of the $2^i n - 1$ sentences already in the extract. To simplify matters, we note that $2^i n - 1$ can never exceed m, so the number of comparisons must be, at worst, proportional to m. The comparison function, $\mathrm{sim}()$, runs in time proportional to the number of word types, w, in the original document. Thus an upper bound on the time complexity of a naïve implementation of Algorithm 1 is $O(wm^2)$.

Running time can be cut down considerably in the general case, however. Since $\mathrm{sim}(i, j)$ remains constant, we can save time by precomputing a triangular similarity matrix of all pairs of sentences in the document, or better yet, by using memoization (i.e., caching intersentential similarity values as they are computed). The algorithm could be further improved by having the loop skip over adjacent extract sentences for which no glue was found on a previous recursion. At any rate, the running time of the summarizer as a whole will likely be dominated by the SVD step of the LSA stage (at least $O(wm^2)$).

3 Evaluation

3.1 *Source data*

We had hoped to use the TIPSTERdocuments commonly used in summary evaluations at the annual Document Understanding Conference (DUC). However, most of them were very short and focussed on single, narrow topics, making them unsuitable for an evaluation of summary coherence. We therefore randomly selected one 1000-word and one 2000-word article from a current encyclopædia, plus one of the five longest newspaper articles from the DUC 2001 trial data.

3.2 *Comparison systems*

On the basis of our own informal observations, we determined that our system (hereinafter `lsa`) performed best with a retention of 20–30% of the singular

values and thresholds of $\alpha = 0.9$, $\beta = 1.0$, and $\gamma = 0.1$. More parsimonious cutoffs tended to result in summaries greatly in deficit of the allowed length.

We selected four third-party comparison systems based on their availability and similarity to our own technique and/or goals: Microsoft Word,commonly available and therefore an oft-used benchmark; Lal & Rüger (2002), a Bayesian classifier summarizer intended to assist students with reading comprehension; Copernic,a commercial summarizer based partly on the work of Turney (2000); and Sinope(formerly Sumatra),which, like `lsa`, employs a technique for identifying latent semantic relations (Lie 1998). In our results tables we refer to these systems as `word`, `plal`, `copernic`, and `sinope`, respectively. In addition to the above systems, we employed random- and initial-sentence baselines (`random` and `init`), as well as a version of our summarizer which does not use the singular value decomposition module (`nolsa`).

3.3 *Test procedure*

We ran the eight summarizers on the three source documents twice each — once to produce a "short" summary (around 100 words) and once to produce a "long" summary (around 300 words). We then recruited human judges who self-identified as fluent in English, the language of the source documents. The judges were provided with these documents and the 48 summaries grouped according to source document and summary length. We asked the judges to read each source document and then assign to each of its summaries an integer score ranging from 1 (very poor) to 5 (very good) on each of three dimensions: comprehensiveness (i.e., topic coverage), coherence, and overall quality. The judges were told only the compression ratio for each summary and told to take it under consideration when assigning their ratings.

4 Results

4.1 *Interjudge agreement*

To compare interjudge agreement, we computed correlation matrices for each of coherence, comprehensiveness, and overall quality ratings. Interjudge agreement on coherence was generally low, with the mean Pearson correlation coefficient r ranging from 0.0672 to 0.3719. Agreement on comprehensiveness and quality was better, but still only moderate, with r in the ranges $[0.2545, 0.4660]$ and $[0.2250, 0.4726]$, respectively. Why the correlation is only moderate is difficult to explain, though given the similarly low agreement in the DUC 2001 evaluations (Lin & Hovy 2002), it was not entirely unexpected. Though we had made an effort to narrowly define coherence in the written instructions to the judges, it is possible that some of them nevertheless conflated the term with its more conven-

Rank(s)		Summarizer	Mean rating
A		`init`	11.1111
A	B	`plal`	9.9722
A	B	`copern`	9.6667
C	B	`word`	8.9444
C	B	`lsa`	8.7222
C	B	`nolsa`	8.6667
C	B	`random`	8.4722
C		`sinope`	7.7500

Table 1: *Summarizer coherence rankings*

tional meaning of intelligibility, or with cohesion. As discussed in Miller (2003), this last possibility seems to be supported by the judges' written comments.

4.2 *Comparative performance of summarizers*

We used SAS to perform a three-way repeated-measures analysis of variance (ANOVA) for each of the three dimensions: coherence, comprehensiveness, and overall quality. Quite unexpectedly, the (*document*, *summary length*, *summarizer*) three-way interaction effect was significant at the 0.05 confidence level for all three dimensions ($p = 0.0151$, $p < 0.0001$, and $p = 0.0002$, respectively). This means it would have been very difficult, if not impossible, to make any generalizations about the performance of the individual summarizers. On the assumption that the type of document was irrelevant to summarizer performance, we added the document scores for each (*summarizer*, *summary length*, *rater*) triplet to get new coherence, comprehensiveness, and overall quality measurements in the range $[3, 15]$. We then performed two-way repeated-measures ANOVAs for each dimension. The two-way interaction effect was still significant for comprehensiveness ($p = 0.0025$) and overall quality ($p = 0.0347$), but not for coherence ($p = 0.6886$).

4.2.1 *Coherence*

In our coherence ANOVA, the only significant effect was the summarizer ($p < 0.0001$). That summary length was not found to be significant ($p = 0.0806$) is somewhat surprising, since we expected a strong positive correlation between the coherence score and the compression ratio. Though we did ask our judges to account for the summary length when assigning their scores, we did not think that very short extracts could maintain the same level of coherence as their longer counterparts. It may be that summary length's effect on coherence is significant only for summaries with much higher compression ratios than those used in our

Short summaries				Long summaries			
Rank(s)		Summarizer	Mean rating	Rank(s)		Summarizer	Mean rating
A		copern	10.0556	A		plal	11.9444
A		plal	9.6667	A	B	copern	10.5556
A	B	init	8.5556	A	B	init	10.2222
A	B	nolsa	8.1111		B	sinope	9.6667
	B	lsa	7.5556		B	word	9.6111
C	B	sinope	7.0000		B	random	9.2222
C	B	word	6.9444		B	lsa	8.9444
C		random	5.3889		B	nolsa	8.9444

Table 2: *Summarizer comprehensiveness rankings*

Short summaries				Long summaries			
Rank(s)		Summarizer	Mean rating	Rank(s)		Summarizer	Mean rating
A		copern	9.7222	A		plal	11.1667
A	B	init	9.4444	A	B	init	10.2778
A	B	plal	9.0556	A	B	copern	9.9444
A	B	nolsa	7.5000	A	B	word	9.2222
C	B	lsa	7.3333	A	B	lsa	9.0556
C		word	6.9444		B	random	8.5000
C		sinope	6.7778		B	nolsa	8.3333
C		random	5.5556		B	sinope	8.1667

Table 3: *Summarizer overall quality rankings*

study.

With respect to the comparative performance of the summaries, only 7 of the 28 pairwise comparisons from our ANOVA were significant at the 0.05 confidence level. The initial-sentences baseline was found to perform significantly better than every other summarizer ($p \leq 0.0008$[3]) except copernicand plal. The only other significant result we obtained for coherence was that the sinopesummarizer performed worse than copernic($p = 0.0050$) and plal ($p = 0.0005$). Using these pairwise comparisons, we can partition the summarizers into three overlapping ranks as shown in Table 1.

4.2.2 *Comprehensiveness and overall quality*

The mean comprehensiveness score for long summaries was higher than that for short summaries by a statistically significant 1.9792 ($p < 0.0001, \alpha = 0.05$). In

[3] All p values from here on are Tukey-adjusted.

fact, in no case did any summarizer produce a short summary whose mean score exceeded that of the long summary for the same document. This could be because none of the short summaries covered as many topics as our judges thought they could have, or because the judges did not or could not completely account for the compression level. In order to resolve this question, we would probably need to repeat the experiment with abstracts produced by human experts, which presumably have optimal comprehensiveness at any compression ratio.

Likewise, the overall quality scores were dependent not only on the summarizer but also on the summary length, but it is not clear whether this is because our judges did not factor in the compression ratio, or because they genuinely believed that the shorter summaries were not as useful as they could have been for their size.

As with coherence, we can partition the summarizers into overlapping ranks based on their statistically significant scores. Because the (*summary length*, *summarizer*) interaction was significant, we produce separate rankings for short and long summaries. (See Tables 2 and 3.)

4.3 *Analysis*

Unfortunately, moderate to low interjudge agreement for all three dimensions, coupled with an unexpected three-way interaction between the summarizers, the source documents, and the compression ratio, stymied our attempts to make high-level, clear-cut comparisons of summarizer performance. The statistically significant results we did obtain have confirmed what researchers in automatic summarization have known for years: that it is very hard to beat the initial-sentences baseline. This baseline consistently ranked in the top category for every one of the three summary dimensions we studied. While the `copern`and `plal` systems sometimes had higher mean ratings than `init`, the difference was never statistically significant.

The performance of our own systems was unremarkable; they consistently placed in the second of the two or three ranks, and only once in the first as well. Though one of the main foci of our work was to measure the contribution of the LSA metric to our summarizer's performance, we were unable to prove any significant difference between the mean scores for our summarizer and its non-LSA counterpart. The two systems consistently placed in the same rank for every dimension we measured, with mean ratings differing by no more than 6%. As a case study in Miller (2003) suggests, this nebulous result may be due more to the LSA summarizer's unfortunate choice of topic sentences than to its gluing process, which actually seemed to perform well with the material it was given.

5 Conclusion

Our goal in this work has been to investigate how we can improve the coherence of automatically-produced extracts. We developed and implemented an algorithm which builds an initial extract composed solely of topic sentences, and then fills in the lacunæ by providing linking material between semantically dissimilar sentences. In contrast with much of the previous work we reviewed, our system was designed to minimize reliance on language-specific features.

Our study revealed few clearly-defined distinctions among the summarization systems we reviewed, and no significant benefit to using LSA with our algorithm. Though our evaluation method for coherence was intended to circumvent the limitations of automated approaches, the use of human judges introduced its own set of problems, foremost of which was the low interjudge agreement on what constitutes a fluent summary. Despite this lack of consensus, we did note a strong ($r = 0.6842$) positive correlation between the judges' scores for coherence and overall summary quality. We would like to take this as good evidence that the production of coherent summaries is an important research area within automatic summarization. However, it may be that humans simply find it too difficult to evaluate coherence in isolation, and end up using other aspects of summary quality as a proxy measure.

Acknowledgments. This research was supported in part by the Natural Sciences and Engineering Research Council of Canada. Thanks to Graeme Hirst and Gerald Penn for their advice.

REFERENCES

Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer & Richard Harshman. 1990. "Indexing by Latent Semantic Analysis". *Journal of the American Society For Information Science* 41:6.391-407.

Lal, Partha & Stefan Rüger. 2002. "Extract-based Summarization With Simplification". *Proceedings of the Workshop on Automatic Summarization, 40th Annual Meeting of the Association for Computational Linguistics* ed. by Pierre Isabelle, vol.II, 90-96. San Francisco: Morgan Kaufmann.

Lie, Danny H. 1998. "Sumatra: A System for Automatic Summary Generation". *Proceedings of the 14th Twente Workshop on Language Technology* ed. by Djoerd Hiemstra et al., 173-176. Enschede: Univ. of Twente.

Lin, Chin-Yu & Eduard Hovy. 2002. "Manual and Automatic Evaluation of Summaries". *Proceedings of the Workshop on Automatic Summarization, 40th Annual Meeting of the Association for Computational Linguistics* ed. by Pierre Isabelle, vol.I, 45-51. San Francisco: Morgan Kaufmann.

Miller, Tristan. 2003. "Generating Coherent Extracts of Single Documents Using Latent Semantic Analysis". Master's thesis, Univ. of Toronto.

Turney, Peter. 2000. "Learning Algorithms for Keyphrase Extraction". *Information Retrieval* 2:4.303-336.

Facilitating Email Thread Access by Extractive Summary Generation

ANI NENKOVA* & AMIT BAGGA**

*Columbia University
**Ask Jeeves, Inc.

Abstract

Email threads are the most common way to represent (archived) discussion groups or mailing lists. Due to the large volumes of such archives, better representation of the threads is required in order to allow the users to find topics of interest and to decide which threads to read. This paper discusses our initial approach to generating thread overviews that serve as indicative summaries for the thread. The overviews give the user a better idea of what is discussed in a related set of email exchanges than the currently used conventions can provide. A relatively large user study on fifty email threads was performed and it confirmed the utility of the proposed approach.

1 Introduction

Mailing lists and discussion groups are becoming increasingly popular. They contain a lot of potentially useful or interesting information (Millen 2000) but finding relevant information might be a daunting task. The main reason for this is that whether reading current postings or past archives, one has few clues what exactly is discussed in an email thread before actually reading all the postings in it. One of the most common representations of mailing list archives is a sequence of the threads in the archive, where each thread is shown as an indented list with subject lines and sender's name, time of posting and number of follow-ups for the message. Even when the subject of the initial posting (root) of the thread is well chosen and informative, the indented representation is not very helpful since the subjects of all follow-up messages are simply "Re: Original Subject". Hypermail[1] and MHonArc[2] are the two most commonly used programs for mailing list archiving and they create browsable representations of the kind just described.

A somewhat better representation is a more hierarchical listing, where the first level contains the subject of the first email in each thread (the root) and the number of postings in the thread and then when a thread/subject is chosen, the indented representation of just this thread is shown in a new window where the messages can be opened and viewed one at a time. The introduction of this extra level gives the user the ability to quickly skim through the initial subject

[1] `http://www.hypermail.org/`
[2] `http://www.oac.uci.edu/indiv/ehood/mhonarc.html`

lines and choose if there is something interesting to read. LISTSERV[3] provides archives of this kind. But, again, even when the subject for the initial posting is well-chosen, it can rarely give the user a good description of what is being discussed.

Google[4] groups further improve on that by using a two-level representation. The first level is similar to the first level in LISTSERV archives and contains date of posting, subject of *thread initial* email, name of most recent poster and number of messages in the thread. The second level consists of a two frame page showing the indented structure and a digest of the messages on one screen. The digest consists of the concatenated bodies of the first ten messages posted in the thread. The Google groups representation therefore allows a user to find topics of interest and to read the threads corresponding to each topic without having to click on every message.

In this paper we describe a more efficient representation that allows a user to decide which threads to read without browsing the actual content of the thread. We call this a thread overviewand it consists of an extractive summary for the documents at the first two levels of the discussion thread tree. The overviews are relatively short and the user can skim through them in order to find threads of interest.

2 Related work

The need for better and easier access to email, discussion groups, and mailing lists has given rise to several directions of research. A substantial body of work addresses the problem of visualization and how it can facilitate access and navigation. Conversation map (Sack 2000), for example, is an interface for discussion archive browsing. The foci in its development are social, semantic and pragmatic aspects of the discussion and how these can be visualized. The system computes links of interaction – who responds to whom, finds "authorities" in the discussion and also creates semantic networks representing similarities and connections between topics discussed in the list. Smith & Fiore (2001) and Donath et al. (1999) also discuss issues with visualizing the huge amount of useful metadata that can be gathered from discussion lists.One of the problems common to all three of these approaches is that representations often involve diagrams, graphs and semantic networks whose interpretation is not intuitive for users.

Work by Muresan et al. (2001) describes an approach to summarizing single emails by key phrase extraction. A corpus of email was manually labeled and machine learning techniques based on linguistic features were applied to obtain a list of noun phrases representing the gist of an email.

[3] `http://www.lsoft.com/products/`
[4] `http://groups.google.com/`

Newman (2002a; 2002b) describes a project about overall archive characterizations and better representation of individual threads. A portion of the essential text of each email is presented in the first level of overview. She also deals with visualization issues and the approach seems promising but no user studies have been conducted so far to evaluate the system.

All the work in the field of multi-document summarization[5] is very related to the problem of creating an overview for an email thread. A lot of research has been done on informative summarization for newswire, but it seems that indicative summarization is what is more needed in the scenario that we discuss. Informative summaries are intended to serve as surrogates for the original document(s), while indicative summaries aim at providing an idea about what is discussed in the document(s) and rather than substituting the original(s) they are supposed to help the user decide if the document(s) are worth retrieving and reading.

3 Task and corpus

Taking into account all the previous discussion on the problem, the current practices and related research, we felt that an indicative summary representation of an email thread can facilitate the retrieval of relevant information from mailing list and mail-based discussion groups. Therefore, we wanted to generate indicative summary representations or overviews of threads that can provide a better idea of the problem discussed than the original subject line alone could give.

Since automated subject/headline generation (Banko et al. 2000) is a hard problem and has been currently addressed only in single document setting, extractive summary generation seemed like a good way to proceed. In other words, extracting one informative sentence per email and using that as the subject would provide a preview of the content of an email. Such an approach complements the approach taken by Newman (2002a) where parts of messages are shown as part of an overview page. An obvious starting point would be to use the first sentence of an email as its subject. However, unlike newswire, first sentences in email can often be greetings, quotes from a previous message, header information, etc., and therefore they, in general, may not be informative. Nevertheless, we used a variant of first sentence extraction as a baseline for comparing the performance of our approach (described further in Section 5).

3.1 *Corpus*

We chose to work with the Pine-Info mailing list which contains emails regarding "features, bugs and workarounds, usage, installation, customization and more

[5] See for example the Document Understanding Conference publications `http://www-nlpir.nist.gov/projects/duc/pubs.html`.

pertaining to the Pine software"[6]. The choice of this list was deliberate as the discussion there is very focused and usually problem-solving oriented and our approach is targeted toward discussion lists of this kind. In comparison, the discussion on general topics can be much more loosely related and far more difficult to process successfully.

The length of discussion paths (a single branch of the discussion tree) and their branching factor (the maximum number of answers a message gets) can vary significantly from one discussion group to another. An analysis of the corpus showed that neither the depth nor the branching factor of the threads were very large. Table 1 shows that paths in threads from the list tend to be short, with the majority including just a root and a follow-up to it (depth 2) and that chains of replies longer than 4 are not typical. Threads in the list are also not very bushy, with most messages receiving just one or two replies. The exact figures can be seen in Table 1.

The shallow, thinly branched threads in this corpus are the result of the focused nature of the list. The initial emails are usually requests for help and they often receive direct answers. This makes the top level of the discussion tree (the root plus the first level of answers) very useful since they often contain statements of problems and their solutions. We should however note that not all threads start with a problem question followed by a series of answers. There are numerous cases where the initial email starts a general discussion, solicits opinions, reports bugs, suggests new features, etc. The archive provided us with 2389 threads to work with.

depth	2	3	4–7	8-21	successors	1	2	3-7
branches	9999	1930	1586	189	messages	12058	2908	746

Table 1: *Length of paths found in threads and their branching factors. On the left are shown the number of branches (paths) with the specified depth. On the right, the number of messages with a given number of successors are listed and leaf messages with no succesors are not shown*

4 Generating thread overviews

As described earlier, our goal was to generate an indicative summary representation of a thread by extracting, from each email in the thread, a sentence which substitutes its subject line. Because of the problem solving nature of the discussion list, we noticed that a further reduction can be made by extracting a sentence only from the thread root message and its immediate follow-ups. These extracts

[6] `http://www.washington.edu/pine/pine-info/`

ideally contain a statement of the problem and a suggestion for its solution, they are easy to read and also they give the user sufficient information on the topic of the thread so that the user can decide if he needs to read the entire thread. Thus, we do not use a fixed compression rate in terms of words for the produced summaries, but rather aim at a more flexible stratagy in which one sentence per message is chosen.

Figure 1 shows an example of an indicative summary of a thread. The first line, in bold, is the original subject line of the root message. Below this line is the summary subject line from the root followed by two summary subject lines from the two replies.

The algorithms used to generate such an indicative summary are described below.

4.1 *Extracting sentences from roots*

Often initial postings have well-chosen subjects. In order to find the sentence in the root email that contains the heart of the problem, rather than background or introduction information, we find the shortest sentence in the email that has the largest overlap of nouns, verbs, adjectives or adverbs with the subject of the message. There are four stages in the algorithm:

(1) Clean any existing quotation and signature blocks from the root.
(2) The message and its subject are processed with LT POS Mikheev (1996). Sentence boundaries and parts of speech are assigned.
(3) Every noun or verb is substituted by its non-inflected lexical form, obtained from WordNet (Miller et al. 1990).
(4) Each sentence is assigned a score equal to $overlap_{subj}/length_{sent}$, where $overlap_{subj}$ is the overlap of noninflected forms of verbs, nouns, adjectives and adverbs in the subject and the sentences, and $length_{sent}$ is the number of words with such parts of speech in the scored sentence. In the case of ties, the sentence with highest score that appears first in the body of the message is chosen.

The sentence with highest score is extracted. The normalization to sentence length was aimed at picking shorter sentences. Very long sentences are not easy to skim and also for such sentences the probability of larger overlap is naturally higher. We will return to the issue of normalizing the sentence length for the root in Section 5.

4.2 *Extracting sentences from follow-ups*

In this stage, the overlap in terms of verbs and nouns between the root message and each of the sentences of the follow-up message is computed.

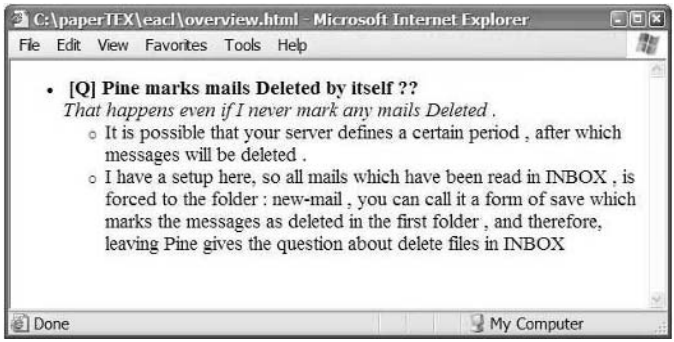

Figure 1: *Thread Overview Example*

(1) Clean any existing quotation and signature blocks in both the root and the follow-up messages.
(2) Both the root mail and its follow-up are processed with LT POS to get part-of-speech tags and sentence boundaries. Using WordNet, nouns and verbs are converted to their noninflected forms.
(3) For each sentence in the follow-up, a weight is computed equal to the overlap of nouns and verbs between the root email and the sentence.
(4) The sentence with highest score is extracted as a "main topic" for the follow-up message. Again, ties are decided by giving preference to the sentence with the highest score that appears earliest in the message.

Thus, the root message is taken as background and the sentence most relevant to the background in the follow-up is extracted. This approach helps make the overviews more cohesive as it ensures that the subjects of the follow-ups are related to the subject of the root.

An example of a thread overview is shown in Figure 1. Figure 2 provides the full text of the messages in the thread.

5 Evaluation and results

Evaluating machine generated summaries in an automated way is currently an unsolved problem. Therefore, we used human subjects to judge the indicative summaries produced by our system. We used fifteen human subjects to evaluate fifty randomly selected threads containing 161 messages total. Each thread was evaluated by three subjects and the majority decision was taken as final.

We asked the users to evaluate the summaries in three dimensions:

	GOOD	BAD
roots	58%	42%
follow-ups	64%	36%
before reading messages	74%	26%
after reading messages	68%	32%

Table 2: *The first two rows show the appropriateness rating for the sentences extracted per email. The last two rows contain the rating of how indicative the entire overviews were both before and after reading the actual thread. GOOD means the evaluator could form an expectation about the subject matter of the message/thread and BAD means the summary was not helpful*

(1) How informative was the overview? How good idea does it give you about what is being discussed in the thread?
(2) Was the summary subject line of the root message appropriate? Can you get from it an idea of what the message is about?
(3) Were the summary subject lines of each of the responses appropriate?

The first question is definitely different from the other two because it is possible that well chosen subject sentences do not make up a good overview, or alternatively, the overview can be informative even when the most appropriate sentences are not chosen as the subject for either the root or one or more of the follow-ups. In order to capture this variation we asked our subjects to first judge threads as a whole and then separately judge the appropriateness of each of the subject sentences with respect to the email it is extracted from. In addition, since it is possible that overviews may suggest a misleading topic of discussion, we asked the human judges to give their opinion on the informativeness of the thread in two scenarios. First, they were asked, before reading any of the messages in the thread, to read the overview and decide if they can form an idea of what will be discussed in the thread. Then, after reading the messages, they gave their opinion again, now acquainted with the actual content of the message. Opinions did change – out of the 150 total judgments, there were 22 changes of opinion where an overview first seen as informative was judged misleading after reading the whole thread, and there were 7 changes in the opposite direction.

Table 2 shows the ratings on the appropriateness of the extracted sentences for each of the messages in the fifty threads. A closer look at the numbers showed that the performance for root messages was lower than that of the follow-ups. A subsequent analysis showed that normalizing the sentence length resulted in concise but not necessarily the most appropriate sentences being picked as the subject. Table 2 also shows the ratings on how indicative the overviews really were. The first observation here is that the numbers for the indicativeness of the overviews are much higher than those for the appropriateness of the sentences.

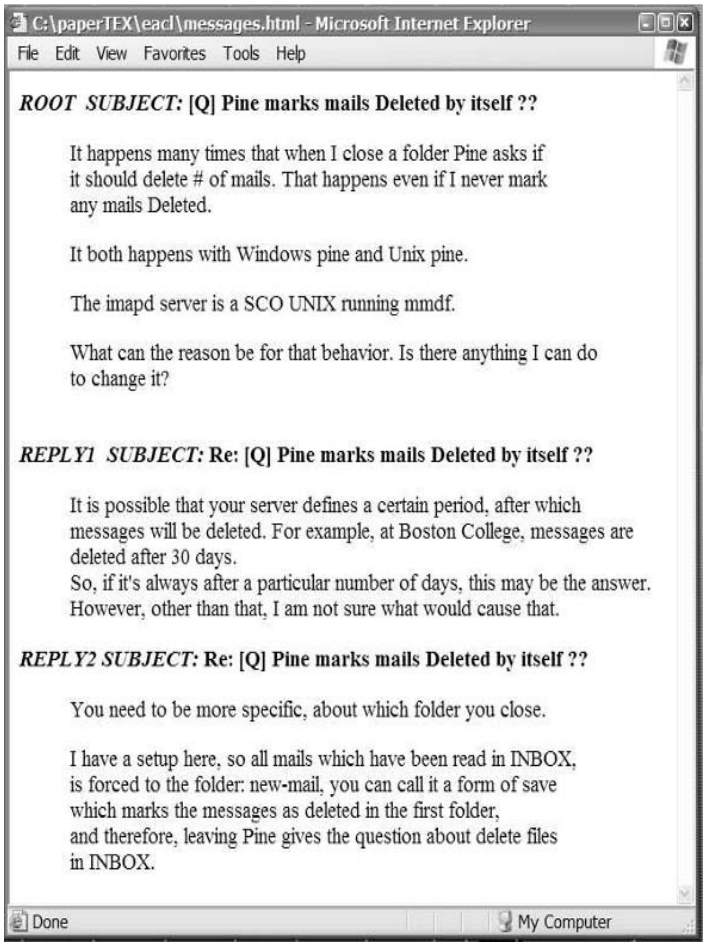

Figure 2: *Messages for which the overview in Figure 1 was generated*

The figure also shows that there was a net 6% change of opinion in the negative direction. While the final numbers in this figure are not in the desired 80% plus range, we feel that when compared to the existing schemes of repeating the subject line of the root messages, our representation provides a huge improvement with 68% of the overviews actually being judged as good.

The judges were also asked to compare the thread overview generated by sentence extraction and a baseline overview generated by using the first sentence

in a message. However, as noted earlier in the paper, the first sentence of a message can consist of quotes from previous messages in the thread, greetings, etc. Therefore, in order to make the comparison realistic, we manually chose the first "useful" sentence of the message. The overview generated by our system was better than or equal to the baseline 74% of the time. The three judges could not form a majority opinion in 10% of the cases while in 16% of the cases they preferred the baseline overview.

6 Conclusions and future work

The user study evaluation shows that extractive techniques can help enhance access to discussion group archives. A natural next step will be to try multi-document summarization approaches to the messages at the same level of the discussion tree. In order to get maximum benefit from the thread overviews, the extraction of sentences needs to be combined with visualization techniques.

REFERENCES

Banko, Michele, Vubhu O. Mittal & Michael J. Witbrock. 2000. "Headline Generation Based on Statistical Translation". *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics* (*ACL'2000*), 318-325. Hong Kong.

Donath, Judith, Karrie Karahalios & Fernanda Viegas. 1999. "Visualizing Conversations". *Proceedings of the 32nd Hawaii International Conference on System Sciences*, Maui, Hawaii.

Mikheev, Andrei. 1996. "Learning Part-of-speech Guessing Rules from Lexicon: Extension to Non-Concatenative Operations". *Proceedings of the 16th International Conference on Computational Linguistics* (*COLING'96*), 237-234. Copenhagen, Denmark.

Millen, David. 2000. "Community Portals and Collective Goods: Conversation Archives as an Information Resource". *Proceedings of HICSS - 33rd Annual Hawaii International Conference on Systems Sciences*, vol. 3., p.3030. Maui, Hawaii. — `www.computer.org/proceedings/hicss/0493/04933/04933030.pdf` [Source checked in May 2004]

Miller, George A., Richard Beckwith, Christiane Fellbaum, Derek Gross & Katherine J. Miller. 1990. "Introduction to Wordnet: An On-line Lexical Database". *International Journal of Lexicography* 3:4.235-312.

Muresan, Smaranda, Evelyne Tzoukerman & Judith Klavans. 2001. "Combining Linguistic and Machine Learning Techniques for Email Summarization". *Proceedings of the Workshop on Computational Natural Language Learning at ACL-EACL 2001* (*CoNLL'2001*) ed. by Walter Daelemans & Rémi Zajac, 152-159. Toulouse, France.

Newman, Paula S. 2002a. "Email Archive Overviews Using Subject Indexes. *Proceedings of the Conference on Human Factors and Computing Systems* (*CHI2002*), 652-653. Minneapolis, Minnesota.

Newman, Paula S. 2002b. "Exploring Discussion Lists: Steps and Directions". *Proceedings of 2nd ACM/IEEE-CS Joint Conference of Digital Libraries*. Portland, Oregon. — `www2.parc.com/istl/groups/hdi/papers/psn_digilib2002.pdf` [Source checked in May 2004]

Sack, Warren . 2000. "Conversation Map: An Interface for Very-large-scale Conversations". *Journal of Management Information Systems* 17:3.73-92.

Smith, Marc A. & Andrew T. Fiore. 2001. "Visualization Components for Persistent Conversations". *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 136-143. Seattle, Washington.

Towards Deeper Understanding of the Latent Semantic Analysis Performance

PRESLAV NAKOV*, ELENA VALCHANOVA** & GALIA ANGELOVA**

*University of California at Berkeley
**Bulgarian Academy of Sciences

Abstract

The paper studies the factors influencing the performance of the Latent Semantic Analysis. Unlike previous related research that concentrates on parameters such as matrix elements weighting, space dimensionality, similarity measure etc., we address the impact of another fundamental factor: the definition of "word". For the purpose, series of experiments were performed on two corpora in order to compare (with respect to the task of text categorisation) six word variants with different linguistic quality. The results show that while the linguistic processing influences the performance, the traditional factors are more important.

1 Introduction

The contemporary *Information Retrieval* (*IR*) relies almost exclusively on *bag-of-words* models and individual words counts. Multi-word terms are less often used as basic language elements due to the complexity and ambiguity of their recognition. Thus, the definition of *word/term* from the point of view of the particular IR algorithm may turn to be of crucial importance. It could be just the surface *word type* (*form*) as seen in the text, the *lemma* (the canonical form) after inflexions removal, the *root* or the *stem* (a prefix shared by the different forms of the same word). In the latter case, a stem can group together a set of inflected forms only (of the same root), but often includes derivational variants as well. In addition, the *homographs* can be further disambiguated: this can be limited to *Part-Of-Speech* (*POS*) only, or may involve *word sense disambiguation* (when a supervised algorithm is used) or *word sense discrimination* (when unsupervised, e.g., clustering). Finally, the *terms* can be multi-word phrases or named entities.

There is a default assumption in the computational linguistics community that a better, linguistically motivated definition of *word* may improve the IR results, whatever *"improvement"* means, but this remains to be proven or at least checked in carefully designed experiments. Our present work is focused on *Latent Semantic Analysis* (*LSA*) in an attempt to better understand the role of the preliminary linguistic processing for text classification of Bulgarian documents. Section 2 briefly introduces LSA as an IR technique. Section 3 comments the related work. Section 4 overviews the resources used. Section 5 describes our

experiments and section 6 discusses the results. Section 7 contains the conclusions and future work.

2 LSA and text categorisation

LSA is among the most popular IR techniques. It assumes an internal structure of the word usage that cannot be observed directly due to the freedom of lexical choice: a variety of words and word combinations can refer to the same notion, as different people use the same words to describe the same object 10-20% of the time only (Furnas et al. 1986). The core idea of LSA is the assumption of mutual latent dependencies between the words and the contexts they are used in (phrases, paragraphs or texts). By taking advantage of this, LSA can deal with synonymy and partially with polysemy, which are among the major problems with the word-based approaches.

LSA is fully automatic and does not need linguistic resources. It is a two-stage process including learning and analysis of the indexed data. On *learning*, LSA performs an automatic document indexing and constructs an $m \times n$ matrix X with columns associated with the n *documents*, and rows with the m *terms*. The matrix X is subject to *singular value decomposition* (*SVD*), which is supposed to remove the unnecessary noise while revealing the latent factors. It compresses the original space in a much smaller one where only a limited number of singular values are kept (typically between 100 and 400; experiments for English show 300 is close to optimal (Landauer & Dumais 1997)). As a result, a vector of reduced dimensionality is associated with each term and with each document. The second phase is the *analysis*, when the proximity between two documents/terms is calculated as the dot product between their normalised LSA vectors.

In our experiments, we built an LSA matrix from the documents in the training set. The new document to be classified is projected in the LSA space and then compared to each one from the training set with cosine as a similarity measure. We used k*-nearest-neighbour* (*k*NN), which is among the best performing text categorisation algorithms: calculate a similarity score between the document to be classified and each of the labelled documents in the training set. When $k = 1$ the class of the most similar document is selected. Otherwise, the classes of the k closest documents are used, taking into account their scores[1]. Finally, in addition to LSA, we considered for comparison the standard *vector-space model* (with no dimensionality reduction).

[1] We can use *direct sum* or *sum after dividing by the rank* of the document in the sorted list of the closest k ones. Consider $k = 6$ and let the 6 closest documents have scores/classes as follows: .98/*cls2*, .76/*cls1*, .65/*cls3*, .53/*cls1*, .47/*cls2* and .33/*cls1*. When just adding the individual scores, we obtain: *cls1*: .76+.53+.33=1.62, *cls2*: .98+.47=1.45 and *cls3*: .65. So *cls1* wins. If we divide by the rank, we have: *cls1*:.76/2+.53/4+.33/6=.5675, *cls2*: .98/1+.47/5=1.074 and *cls3*: .65/3=.2167, so *cls2* wins.

3 Related work and motivation

A variety of algorithms have been applied to supervised text categorisation: Naive Bayes, k-nearest-neighbour (*k*NN), Rocchio, support vector machines, decision trees, decision lists, neural networks, maximum entropy, expectation maximisation, linear least squares etc., see (Yang 1999) for an overview. For application of LSA to text categorisation see (Bartell & al. 1992; Foltz & Dumais 1992). LSA is highly dependent on the parameters tuning (Nakov 2000), see (Dumais 1991) and (Nakov et al. 2001) for a deeper study of the weight functions impact on LSA performance. (Nakov et al. 2003) contains a longer list of relevant work, dealing with application of LSA to text categorisation and parameters tuning. Here we briefly point to the following:

1. Although LSA is a comparatively old and well-studied technique, its effective usage requires sophisticated tuning, which is viewed as a kind of art. Some of the most important performance factors are: (*i*) definition of term, (*ii*) matrix elements weighting, (*iii*) space dimensionality, and (*iv*) similarity measure. LSA has been used for a variety of tasks, including text categorisation. We found the latter more natural for automatic evaluation since it allows relatively easy results comparison and is to certain extent more "objective" than the classic IR. There are several subtasks in text categorisation: (*i*) topic identification, (*ii*) authorship attribution, (*iii*) text genre classification, (*iv*) language identification, etc. The experiments reported below are restricted to *topic identification*.

2. The definition of *term* in LSA input texts attracted less attention so far and has not been investigated systematically as a performance factor. To the best of our knowledge, it is not studied at all for the highly inflectional Slavonic languages, which motivates our current research.

3. Stemming for Bulgarian is an interesting problem because of the *definite* and *indefinite articles*, which appear *augmented* at the very *end* of the words. We used a rule-based inflectional stemmer for Bulgarian (Nakov 2003) with stemming rules learned automatically from the morphological dictionary we have at hand. In order to improve the coverage, we applied all the 93,066 stemming rules for the three-letter left contexts.

4 Linguistic resources and text collections

A key resource for the experiments reported below is the large *Morphological Dictionary of Bulgarian*, created at the Bulgarian Academy of Sciences. The texts from the corpus were lemmatised according to this dictionary and the list of 442 stopwords was derived from it as well. Although the stemmer targets the *inflectional* morphology only, the resulting stems sometimes conflate different *derivational* variants as well. This means, sometimes it is potentially more powerful than lemmatisation (although it is conservative and not as aggressive as the

traditional stemmers for English, e.g., the Porter stemmer (Porter 1980)). From a linguistic perspective, stemming is less "correct" than lemmatisation, so we wanted to compare them.

We built a special collection (called *Set15*) of news articles from Bulgarian online sources, which includes 702 different documents manually grouped in 15 categories. *Set15* contains 19,429 word types and 406,783 word tokens (the numbers and the non-Cyrillic symbols are excluded). Further, we filtered out all single letter words as well as the ones met in a single document (as they cannot contribute to the similarity between two documents when cosine is used as a similarity measure). As a result, the word types count dropped to 19,301 and the word tokens count – to 259,000. In addition, we built another collection *Set4* (127 documents), a subset of the intial one, containing the documents from 4 categories only: (*i*) *Agriculture&Foresty* – 12 documents, (*ii*) *Culture* – 33, (*iii*) *Defence* – 15 and (*iv*) *Sport* – 67. When the filtering was applied to *Set4* the word types/tokens dropped from 15,483/80,016 to 5,530/36,527. *Set4* contains 4,487/71,879 stems for types/tokens respectively, obtained using the above-mentioned stemmer, 4,558/73,018 lemmas, 951/1,455 phrasal terms and 190/458 named entities. Semantic processing concerned named entities only, as synonymy of institution names was relevantly marked. A more detailed discussion regarding the data can be found in (Nakov et al. 2003).

A potential problem with this stopwords removal is that many of the stopwords can be regular ones depending on their POS. There are 53 such stopwords in our dictionary. So, for the POS disambiguation, lemma and lemma&phrase experiments described below we checked the POS before filtering. For the stemming experiments no checking was performed.

5 Experiments and evaluation

The evaluation was based on a *stratified 10-fold cross-validation*. For the purpose, *Set15* was split into 10 sets of almost equal size such that the class distribution in each set follows as close as possible the class distribution in the original collection. We ran 10 tests, each time training on 9 folds and testing on the remaining one. We calculated the classification accuracy on the test set for each run and took the average over the 10 runs.

Set4 is smaller and we did not want to lose too much data for training, so we performed a stratified *20-fold* cross-validation. An alternative would be to follow a *leave-one-out* strategy: train on 126 documents and test on the remaining one. But since we remove a document from one class only, this class will suffer, unlike the rest, and thus the results will not be a good approximation of the real performance. We decided that 20 is a good compromise between the need to model to some extent the original distribution and to waste as less documents as possible during testing.

As we will see below, the choice of weighting functions, applied prior to SVD can have a dramatic impact on the further performance. The weighting is expressed in terms of the *Local* and *Global Weight Functions* (*LWF* and *GWF*). The LWF $L(i,j)$ represents the weight of term i in document j, while the GWF $G(i)$ expresses the weight of term i across the entire document collection. Some popular weighting schemes for LSA, together with their numerical codes used in our tables, follow (Nakov et al. 2001):

- LWF = 0: $L(i,j) = x(i,j)$ − frequency of term i in document j
- LWF = 1: $L(i,j) = \log(1 + x(i,j))$ − logarithm
- GWF = 0: $G(i) = 1$ − trivial
- GWF = 1: $G(i) = \frac{1}{\sqrt{\sum_j L(i,j)^2}}$ − normalised
- GWF = 2: $G(i) = g(i)/d(i)$ − GfIdf
- GWF = 3: $G(i) = 1 + \log(N/d(i))$ − Idf
- GWF = 4: $G(i) = -\sum_j p(i,j) \log p(i,j)$
- GWF = 5: $G(i) = 1 + \frac{\sum_j p(i,j) \log p(i,j)}{\log N}$

where: N is the training documents count, $g(i)$ − frequency of term i across all documents, $d(i)$ − number of documents containing i, $p(i,j)$ − probability (normalised frequency) of observing term i in document j.

Both GWF=4 and GWF=5 represent some kind of entropy. GWF=4 performed slightly worse than GWF=5 but otherwise exhibited the same behaviour, so we removed it from Tables 1 and 2 in order to save space.

We performed a large number of experiments: $11 \times 2 \times 6 \times 5 = 660$, for *Set4* (word definitions with and without stopwords removed, GWF, LWF, LSA dimensions), and $4 \times 2 \times 6 \times 5 = 240$, for *Set15*. For each of these combinations we tried all possible values of k for the *k*NN classifier: ∼114 for *Set4*, and ∼670 for *Set15*. Further, each possibility was tried with and without dividing by the rank (Nakov et al. 2003). So, for *Set4* we ran $11 \times 2 \times 6 \times 5 \times 114 \times 2 = 150,480$ single tests and for *Set15* we had $4 \times 2 \times 6 \times 5 \times 670 \times 2 = 312,600$.

The evaluation results for *Set4* using *k*NN with $k = 1$ are summarised in Table 1, which shows the classifier's micro-average accuracy over the 20 cross-validation runs. Columns 1 and 2 contain the LWF and GWF numerical codes, as described above. Column 3 shows the LSA space dimensionality (*orig.* means: no dimensionality reduction, i.e., the original vector-space, which is 15,438 for raw words; 4,487 for stems etc.). Each row corresponds to a particular combination of LWF*GWF and LSA dimensionality reduction. Columns $4-8/10-14$ summarise the classification correctness for different variants of the input text: *raw*, with *stem*-ed words, with *lemma*-tised words, *l.&ph.* (lemma and phrase together), *POS* (disambiguation in terms of POS. Column 9, *phrase only*, gives results when only multi-word phrases and named entities are used as features.

6 Discussion

The most interesting observations are:

1. The choice of weighting scheme is among the most important factors. When the appropriate combination of LWF*GWF is used (1*3, 1*5, 0*3, 0*5), other factors such as *stopwords removal* and *LSA dimensionality reduction*, become almost irrelevant.

2. Stemming and lemmatisation are almost equally good for the highly inflectional Bulgarian language.

3. For the best performing combination of LWF*GWF (1*3 and 1*5) *the definition of word becomes irrelevant*. Note that this might be due to the particular task (text categorisation), and may not hold for IR in general. Thus, more tests with respect to a variety of IR related tasks are needed. However, if we look at the worse weighting schemes in Table 1, e.g., GWF $\{0,1,2\}$ and *stopwords kept*, we can see that stemming delivers up to 26% improvement. This suggests it could still be important but its contribution could just have been obscured by the impact of weighting. This hypothesis though fails for *Set15*: as Table 2 shows, the impact of stemming is less than 1% in case of 1*3 and 1*5 (see the original space and the LSA dimensionality of 100; 100 is among the most popular dimensionalities and proved to be among the best performing on *Set15*).

4. The stopwords removal makes a really dramatic difference but only when GWF $\in \{0, 1, 2\}$. When the best weighting schemes are used (1*3 or 1*5), its impact is less than 1.5%.

5. With the exception of 1*3 and 1*5, the POS disambiguation performs consistently worse than the other experiments. Even though words with different POS have different meanings, these might be close enough (polysemy) so that it is better to keep them together. This is consistent with the observations of Krovetz that, while resolving homography is beneficial, disambiguating polysemy damages the IR performance (Krovetz 1993).

6. Using *phrases only* is consistently far worse than stopwords removal.

7. Combining phrases and lemmatisation gives slightly improved results over lemmatisation only, but these are still almost indistinguishable from the ones obtained using stemming.

8. LWF$= 1$ leads to substantial benefits but only when GWF $\in \{0, 1, 2\}$ with stopwords kept. When removed, the impact is insignificant. This is easy to explain: the stopwords are the most frequent words and the logarithmic weighting (LWF$= 1$) makes a big difference mostly for them.

9. Changing the number of neighbours k in kNN leads to improvement but only unless an appropriate GWF is used (e.g., 3 or 5). In the latter case the results are already fairly good for 1-NN, so there is no much space for improvement left. While $k = 10$ was among the best performing values for *Set4*, for the bigger *Set15* we obtain a consistent improvement as k grows from 1 up to 40.

L G LSA WW dim F F	STOPWORDS KEPT raw	stem	lemma	l.&ph.	POS	phrase only	STOPWORDS REMOVED raw	stem	lemma	l.&ph.	POS
0 0 10	78	88	84	85	69	83	92	92	96	96	86
0 0 20	82	85	85	85	81	82	92	98	99	99	89
0 0 40	85	86	86	86	85	83	96	100	98	99	92
0 0 orig.	74	89	85	85	72	37	96	96	98	99	91
0 1 10	76	89	81	84	70	83	96	98	96	97	89
0 1 20	85	87	86	86	80	83	95	99	98	98	88
0 1 40	81	89	85	84	77	83	95	98	98	98	93
0 1 orig.	61	87	85	84	66	34	96	96	98	99	91
0 2 10	55	61	65	64	56	81	92	94	93	95	82
0 2 20	56	66	70	70	59	83	92	95	93	95	84
0 2 40	58	67	70	71	59	84	92	98	98	98	86
0 2 orig.	57	68	72	72	59	37	93	98	98	98	86
0 3 10	95	98	99	99	89	83	97	98	99	99	94
0 3 20	96	100	100	100	96	83	99	99	99	99	96
0 3 40	92	99	99	100	94	84	99	100	100	100	97
0 3 orig.	92	98	96	97	91	37	99	100	100	100	98
0 5 10	97	98	99	99	92	84	96	99	99	99	92
0 5 20	98	100	99	99	96	83	98	100	100	100	96
0 5 40	98	100	100	99	96	84	98	100	100	100	96
0 5 orig.	96	100	99	100	94	35	99	100	100	100	96
1 0 10	96	95	96	96	93	82	94	96	97	97	95
1 0 20	90	97	97	97	92	83	98	100	100	100	96
1 0 40	92	96	96	96	88	84	96	98	99	99	96
1 0 orig.	90	94	95	95	90	33	96	98	99	99	96
1 1 10	92	96	96	97	90	83	96	98	97	97	94
1 1 20	89	94	93	94	88	83	93	96	99	99	96
1 1 40	81	89	88	88	83	83	95	94	95	96	96
1 1 orig.	62	85	90	90	80	35	95	92	96	97	88
1 2 10	84	89	88	89	74	81	93	96	96	96	89
1 2 20	87	88	88	89	76	83	94	98	96	96	88
1 2 40	82	88	88	88	77	84	93	98	97	97	88
1 2 orig.	82	93	92	92	82	40	96	98	98	98	91
1 3 10	97	99	99	100	98	83	99	99	99	99	99
1 3 20	99	99	99	99	98	83	98	99	99	99	99
1 3 40	99	99	99	99	99	84	99	100	99	99	99
1 3 orig.	98	100	99	100	97	35	99	100	100	100	99
1 5 10	98	99	99	99	99	83	99	99	99	99	99
1 5 20	99	99	99	99	99	83	98	99	99	99	99
1 5 40	99	99	99	99	99	84	99	100	99	99	98
1 5 orig.	98	100	100	100	99	37	99	100	100	100	100

Table 1: *Set4 – micro-averaging categorisation accuracy for 1-NN*

10. For *Set15*, stemming is beneficial, but the stop-words removal has an even bigger impact. This is not obvious from Table 1 as the values are too close to 100.00%. However, the most important parameter remains the weighting scheme: when GWF $\in \{3, 5\}$ the impact of stemming, stop-words removal and even of LWF is limited.

11. The categories count is another important parameter: going from 4 to 15 categories lowers the best accuracy for *raw words* and *stemming* from 100.00% (Table 1) to 75.50% and 76.35% (Table 2).

12. It is interesting to compare the original vector space and the reduced LSA one. These are often comparable (almost equal for e.g., 1*3 and 1*5) but sometimes there are bigger differences, e.g., for 0*1, stop-words kept, raw words:

61.42% vs. 85.04% (LSA dim. 20). The impact is even more dramatic for *phrase only* experiments: for any combination of LWF*GWF the performance of LSA (for all dimensionality reductions listed) is more than twice the performance without dimensionality reduction: e.g., 33.86% vs. 84.25%. This is explained by the fact that the average number of multi-word phrases per document is very limited, and thus the documents share very few of them, which is not enough to judge similarity in a reliable way. When using LSA though, the projected space contains a summary of the co-occurrences as well as some transitive implications, observed globally across documents. So, even though the LSA vector dimensions are much fewer, they contain much less zero components and thus discriminate better. A similar effect (with a similar explanation) is observed for the raw text without stop-words removal, where LSA proves consistently better (especially for 0*1 and 1*1) for all dimensionality reductions.

LWF	GWF	LSA dim	Stopwords kept		Stopwords removed	
			raw text	stemmed	raw text	stemmed
0	0	20	41.45%	50.14%	60.83%	62.68%
0	0	100	50.00%	59.54%	66.67%	70.66%
0	0	orig.	45.73%	58.55%	69.94%	72.79%
0	1	20	40.17%	51.99%	61.54%	62.25%
0	1	100	46.87%	57.12%	65.95%	71.08%
0	1	orig.	45.58%	58.12%	69.94%	72.93%
0	2	20	26.07%	27.64%	49.86%	58.26%
0	2	100	26.78%	28.63%	60.97%	67.09%
0	2	orig.	27.49%	29.91%	66.10%	70.94%
0	3	20	62.39%	64.67%	68.09%	69.94%
0	3	100	69.09%	73.08%	73.22%	75.21%
0	3	orig.	71.37%	74.36%	75.21%	75.78%
0	5	20	65.53%	69.23%	66.67%	70.23%
0	5	100	72.36%	75.93%	74.07%	76.35%
0	5	orig.	73.79%	75.50%	75.50%	75.93%
1	0	20	60.68%	65.81%	63.96%	66.95%
1	0	100	64.81%	69.37%	71.37%	71.79%
1	0	orig.	69.23%	71.79%	71.37%	72.93%
1	1	20	59.12%	64.10%	64.81%	65.95%
1	1	100	60.40%	66.24%	54.27%	67.38%
1	1	orig.	66.67%	71.79%	71.23%	72.79%
1	2	20	38.46%	50.14%	62.54%	65.67%
1	2	100	47.15%	58.12%	66.81%	71.51%
1	2	orig.	50.28%	60.83%	70.94%	73.22%
1	3	20	68.52%	69.23%	68.66%	69.52%
1	3	100	72.93%	74.07%	74.36%	73.79%
1	3	orig.	72.51%	72.93%	72.22%	73.22%
1	5	20	68.66%	70.66%	68.80%	71.08%
1	5	100	73.36%	75.21%	74.36%	74.93%
1	5	orig.	72.79%	73.08%	72.51%	73.08%

Table 2: *Set15 – micro-averaging categorisation accuracy for 10-NN*

7 Conclusions and future work

Some earlier research on the impact of the linguistically motivated text indexing on IR performance shows that a better word pre-processing is not necessarily

needed for effective retrieval, see (Spark-Jones 1999). It is not clear whether this is due to the weakness of the linguistic analysis, which still does not produce a semantically motivated concept-based representation, or the linguistic analysis as such is not quite relevant for text classification (Peng et al. 2003). The current work allows us to gain important insights regarding the need and the role of the linguistic pre-processing (at least for Slavonic languages). Our experiments show that while the definition of word influences the LSA performance, the traditional IR factors are even more important. It is worth mentioning though that the feature engineering is both language- and task-dependent: e.g., the stopwords may be among the best features for other text categorisation tasks, e.g., language identification or authorship attribution. In the latter case, other features, like linguistic style markers, are usually even more important.

Our experiments show that the word definition becomes almost irrelevant once we stick to topic identification, limit ourselves to the bag-of-words model, work with only few well-separated categories (*Set4*) and use the best weighting schemes (e.g., 1*3 and 1*5). In case of more and less distinguished classes (*Set15*) both stopwords removal and stemming become important (although still far less than the weighting scheme). More experiments are needed to study the impact of lemmatisation, phrases and POS disambiguation in the latter case.

Our future plans include evaluation of the impact of lemmatisation, phrases and POS disambiguation on *Set15* and bigger collections, which would make our experiments and results more complete. It would be also interesting to try using *word senses* (e.g., with respect to some semantic network) instead of the words themselves, but the bad performance of the POS disambiguation above leaves us sceptical. Another reason for scepticism are the negative results already obtained for English: using WordNet senses did not lead to a significant improvement on the *Annotated Brown Corpus* (Kehagias et al. 2001). The problem with the latter work (as well as with the POS disambiguation above) though, may be that, in addition to *homonymy*, the *polysemy* has also been addressed. This is just the contrary to what stemming does. Two other important parameters are missing from our study: *choice of similarity measure* and *feature selection*. Both have been found important for text categorisation. It is also worth trying other classifiers, e.g., Rocchio, Naive Bayes, decision trees etc.

Finally, we would like to consider other IR tasks in an attempt to better understand whether the appropriate word definition is more important for the classic IR than for text topic identification. This would also allow us to perform a comparison with a recent work by Peng et al. (2003), who show that using simple language- and task-independent character-level (as opposed to word-level) models can achieve state of the art results on a variety of text classification tasks.

REFERENCES

Bartell, Brian, Garrison Cottrell & Richard Belew. 1992. “Latent Semantic Indexing as

an Optimal Special Case of Multidimensional Scaling". *15th Conference of the Association for Computing Machinery* (*ACM*) *Special Interest Group on Information Retrieval* (*SIGIR*), 161-167. Copenhagen, Denmark.

Dumais, Susan. 1991. "Improving the Retrieval of Information from External Sources". *Behavior Research Methods Instruments & Computers* 23:2.229-236.

Foltz, Peter & Susan Dumais. 1992. "Personalized Information Delivery: An Analysis of Information Filtering Methods". *Proceedings of Communications of the Association for Computing Machinery* (*CACM*) 35:12.51-60.

Furnas, George, Thomas Landauer, Louis Gomez & Susan Dumais. 1983. "Statistical Semantics: Analysis of the Potential Performance of Keyword Information Systems". *Bell Syst. Tech. Journal*. 62:6.1753-1806.

Kehagias, Athanasios, Vassilios Petridis, Vassilis Kaburlasos & Pavlina Fragkou. 2003. "A Comparison of Word- and Sense-Based Text Categorization Using Several Classification Algorithms". *Journal of Intelligent Information Systems* 21:3.227-247.

Krovetz, Robert. 1993. "Viewing Morphology as an Inference Process". *16th Conf. of the Association for Computing Machinery* (*ACM*) *Special Interest Group on Information Retrieval* (*SIGIR*), 191-202. Pittsburgh, U.S.A.

Landauer, Thomas & Susan Dumais. 1997. "A Solution to Plato's Problem: The LSA Theory of Acquisition, Induction and Representation of Knowledge". *Psychological Review* 104:2.211-240.

Nakov, Preslav. 2003. "BulStem: Design and Evaluation of Inflectional Stemmer for Bulgarian". *Proceedings of the Workshop on Balkan Language Resources and Tools* (*1st Balkan Conference in Informatics*). Thessaloniki, Greece. — `www.iit.demokritos.gr/skel/bci03_workshop/pages/programme.html`

Nakov, Preslav, Elena Valchanova & Galia Angelova. 2003. "Towards Deeper Understanding of the LSA Performance". *Proceedings of Recent Advances in Natural Language Processing* (*RANLP'03*), 311-318. Borovets, Bulgaria.

Nakov, Preslav, Antonia Popova & Plamen Mateev. 2001. "Weight Functions Impact on LSA Performance". *Proceedings of Recent Advances in Natural Language Processing* (*RANLP'01*), 187-193. Tzigov Chark, Bulgaria.

Nakov, Preslav. 2000. "Getting Better Results with Latent Semantic Indexing". *Proceedings of Students Presentations at the European Summer School on Logic, Language and Information* (*ESSLLI'00*), 156-166. Birmingham, U.K.

Peng, Fuchun, Dale Schuurmans & Shaojun Wang. 2003. "Language and Task Independent Text Categorization with Simple Language Models". *Proceedings of the Human Language Technologies Conference* (*HLT-NAACL'03*), 110-117. Edmonton, Canada.

Porter, Martin. 1980. "An Algorithm for Suffix Stripping". *Program*. 14:3.130-137.

Sparck-Jones, Karen. 1999. "What is the Role of NLP in Text Retrieval?" *Natural Language Information Retrieval* ed. by T. Strzalkowski (= *Text, Speech and Language Technology*, 7), 1-24. Dordrecht: Kluwer Academic.

Yang, Yiming. 1999. "An Evaluation of Statistical Approaches to Text Categorization". *Journal of Information Retrieval* 1:1/2.67-88.

Automatic Linking of Similar Texts Across Languages

BRUNO POULIQUEN, RALF STEINBERGER & CAMELIA IGNAT

European Commission – Joint Research Centre

Abstract

Cross-lingual Document Similarity calculation (CLDS) is useful for the navigation of large multilingual document collections and for clustering and classifying documents together independently of their language. We achieve CLDS by mapping documents onto the multilingual EUROVOC thesaurus, which will soon exist in 21 languages, and by representing each document in this multilingual vector space so that the similarity of texts written in different languages can be calculated. An evaluation showed that the system successfully identifies document translations in a large text collection. To adapt the method to the analysis of large multilingual news collections, we combined the mapping with cluster analysis and named entity recognition.

1 Introduction and motivation

In recent years, companies and academia have developed a large number of multilingual text analysis tools. The number of existing *cross-lingual* applications, however, is very limited.Cross-lingual applications establish links between texts written in different languages or give information in one language on a text written in another language. Most cross-lingual efforts are currently spent on Machine Translation and Cross-lingual Information Retrieval (CLIR), i.e., multilingual retrieval of documents following a monolingual search. As the need to access and analyse multilingual document collections in the European Commission (EC) is high, the EC's Joint Research Centre (JRC) has worked on another type of cross-lingual application: *Cross-Lingual Document Similarity* (CLDS) calculation. It allows to establish, for any given document pair written in any of the eleven official languages of the European Union (EU), to what extent the two documents are similar. Two texts that are translations of each other have a very high similarity; two documents about the same subject can automatically be identified as being similar, and two texts talking about completely different subject areas are recognised by showing a very low CLDS value. Similarity values do not differ significantly for texts written in the same or in different languages. A look at the activities of the *Cross-Lingual Evaluation Forum* (Peters 2003) shows that, outside the JRC, CLDS is not an active research area.

CLDS can be used for several purposes: Users that have identified an interesting document may want to read more documents about the same subject area.

They may be particularly interested in reading information about the same subject or event in *different* languages, for instance when analysing the news reporting in different countries about the same subject. In another setting, the aim may be to identify similar texts in order to *avoid* reading more texts about a known subject so as to concentrate on documents covering *other* subjects. CLDS can also be used for multilingual document clustering, i.e., grouping of similar documents written in different languages, or for multilingual document classification, i.e., assigning documents written in different languages to the same content classes.

Our approach to CLDS is based on the automatic mapping of documents onto the fine-grained multilingual classification scheme EUROVOC (Eurovoc 1995), which is used by many parliamentary documentation centres in Europe. The mapping process onto the several thousand EUROVOC classes, which was described in detail in Pouliquen et al. (2003a), consists of identifying a ranked list of the 100 most pertinent content classes (descriptors). In order to identify the best descriptors to represent document contents, we devised an associative systemthat learns large lists of words that are associated to each descriptor term from a training collection of texts to which descriptors had been assigned manually by professional library indexers.A large overlap of a descriptor's associated words and the words of a text indicates that the descriptor is of some relevance to the text.

Section 2 discusses related work. Section 3 briefly describes the EUROVOC thesaurus and summarises the mapping process, as well as the challenges we had to face. Sections 4 and 5 provide details on the way we calculate CLDS and describe how we apply CLDS to allow users to navigate a large multilingual collection of news articles. Section 6 points to future work.

2 Related work

The translation of a document should ideally be the most similar document to a given one. In Pouliquen et al. (2003b), we therefore test CLDS by verifying whether the system successfully identifies the translation of a given document as the most similar one amongst up to 1600 different documents. In the same publication, we refer to a number of different approaches to identify document translations. However, unlike our own more semantically oriented approach, these mainly use information such as formatting features, paragraph length information, HTML anchors, etc. These approaches are thus restricted to the identification of translations and are not suitable for CLDS calculation.

We are aware of one approach that can be used for CLDS calculation: Landauer & Littman (1991) applied cross-lingual *Latent Semantic Indexing* (LSI) to text segments, i.e., smaller parts of a text. The basic function of cross-lingual LSI is to analyse a training set of parallel texts statistically to derive a common vector space representation scheme for all the words in the bilingual sample. Each piece of text in either of the two languages can then be represented using

this same bilingual vector space so that a cosine similarity score can be calculated for each document pair in these two languages. In Landauer & Littman's evaluation on a collection of 1,582 English-French parallel text paragraphs, the average similarity of the translation pairs was 0.78. Their system managed to find the French translation as the most similar text to a given English paragraph in 92% of all cases (92% precision), which is a good result. Although the LSI approach is rather different from our thesaurus-based one, the results achieved are very similar to the ones we achieved. The advantage of Landauer & Littman's approach is that their system can theoretically be trained for any language pair or set of languages, while our own is limited to the languages for which EUROVOC and pre-indexed training material exists. The advantage of our own approach is that it is more modular and that no language-pair specific training is required because EUROVOC provides the link between the languages once the descriptors have been assigned. This is particularly important in our case, as we currently work with eleven languages (55 language pair combinations!) and are aiming to cover all 21 future official languages of the European Union(210 language pair combinations!). Landauer & Littman's approach needs retraining each time a language is added or the training set changes. Another advantage of our approach is that the representation of texts by their most appropriate EUROVOC descriptors fulfils many more uses than CLDS calculation. These include subject-specific summarisation, conceptual and cross-lingual indexing for human users, and automatic annotation of documents for the Semantic Web (Pouliquen et al. 2003a).

3 Mapping documents to the EUROVOC thesaurus

EUROVOC is a hierarchically organised controlled vocabularythat was developed and is used by the European Parliament and many other national and international organisations in and outside the European Union (EU), for the classification, search and retrievalof their large multilingual document collections (Eurovoc 1995). Currently, the cataloguingis a manual (intellectual) process that requires 15 to 20 minutes per document. EUROVOC is currently available in the eleven official EU languages, with an additional ten language versions in preparation. Its 6075 descriptor terms are organised into 21 fields and hierarchically structured with a maximum depth of eight levels. Available relations are broader terms (BT), narrower terms (NT) and related terms (RT). Due to its wide coverage represented by the relatively small number of descriptors, the descriptor terms are mostly rather abstract multi-word expressionsthat are unlikely to be found verbatim in the texts. EUROVOC examples are PROTECTION OF MINORITIES; FISHERY MANAGEMENT and CONSTRUCTION AND TOWN PLANNING.

The procedure of mapping texts written in different languages automatically onto the multilingual EUROVOC thesaurus is described in detail in Pouliquen et al. (2003a) so that this process will only be sketched here. It is

not possible to base an automatic EUROVOC thesaurus descriptor assignmenton the actual occurrence of the descriptor text in the document because the lexical evidence is weak and even misleading: The descriptor text occurs explicitly in only 31% of documents manually indexed with this descriptor. On the other hand, in approximately nine out of ten documents, the descriptor text is present explicitly even though the descriptor itself had not been assigned manually. Basing the assignment on the presence of the descriptor text in the document thus leads to low recall(31%) and precision(7%) values.

For this reason, we have taken another, fuzzy approach. For each descriptor, we automatically produce its profile, i.e., a ranked set of words that are statistically related to the descriptor, so that we have more lexical evidenceat hand that indicates that a certain descriptor should be assigned to a text. As these words are statistically, but not always semantically related, we refer to these pertinent words as descriptor *associates*.We produce these associate lists on the basis of a large collection of manually indexed documents (the *training set*), by comparing the word frequencies in the subset of texts that have been indexed manually with a certain descriptor with the word frequencies in the whole training set. For the comparison, we use a combination of Dunning's statistical log-likelihood test (or G^2) to reduce the number of words to be considered (dimensionality reduction) with filters and various IDF (Inverse Document Frequency) weightings (Salton & Buckley 1988). This method automatically produces lists of typical words for each descriptor and information on the degree with which these words are typical (their weight). The 16 most important associates for the descriptor WILD LIFE, for instance, are: wild_fauna 15; flora 14; wild_bird 11; wild_animal 10; natural_habitat 9; wildlife 9; endanger_species 8; species 6; conservation 6; wild 6; fur 5; humane_trap 5; specimen 5; trap 5; trapping 5; repopulation 5. Associate lists are between 10 and 400 words long.

Before applying any statistical procedures, we carry out some minimal linguistic pre-processingin order to normalise all texts: We lemmatisetexts so as to work only with base word forms, and we mark up the most frequent compoundsin our training set. There is also a large list of stop words, i.e., words that should never be associates because they are not meaningful.

When we want to index a new text automatically with EUROVOC descriptors, we make a statistical comparison of the frequency list of its lemmas with the associate lists of all descriptors to check which associate lists are most similar to the text's lemma list. The most similar associate lists, according to some statistical similarity measure, indicate the most appropriate descriptor terms. The result is thus a long ranked list of EUROVOC descriptors assigned to this document. Typically, we keep the highest-ranking 100 descriptors. The example in Table 1

Rank	Descriptor	Cosine
1	VETERANARY LEGISLATION	42.4%
2	PUBLIC HEALTH	37.1%
3	VETERANARY INSPECTION	36.6%
4	FOOD CONTROL	35.6%
5	FOOD INSPECTION	34.8%
6	AUSTRIA	29.5%
7	VETERANARY PRODUCT	28.9%
8	COMMUNITY CONTROL	28.4%

Table 1: *Assignment results (top 8 descriptors) for 'Food and Vetenary Office Mission to Austria' web document*

shows the assignment results of a document found on the internet.[1]

To speak with text classification terminology (Sebastiani 1999), the mapping of texts onto the EUROVOC thesaurus is a category ranking classification task using a machine learning approach, where an inductive processbuilds a profile-based classifier by observing the manual classification on a training set of documents with only positive examples.For each EUROVOC descriptor, we built a category profile consisting of a set of lemmas and their weight. Unlike usual classifiers, the system does not decide against the appropriateness of a class. Instead, it produces long ranked lists of more or less relevant classes for each document. This representation is more suitable for document similarity calculation.

Specific challenges of the EUROVOC assignment task are: (1) the training documents are multiply classified; (2) due to some frequent descriptor combinations in the training material (e.g., MAURITANIA and FISHERY AGREEMENT), it is at times very difficult to establish different profiles for such descriptors; (3) the amount of training material for the various descriptors is extremely uneven, ranging from 0 and 2964 training documents per descriptor with an average of 73 documents per descriptor and a standard deviation of 181; (4) for most descriptors, the training set contains only a few positive examples, whereas there are about 30000 negative examples,i.e., all documents that were *not* indexed with the descriptor; (5) the descriptor usage distribution is uneven over the different text types of the training collection so that text type-specific features interfere with content features.

According to Sebastiani (1999), the k-nearest-neighbour (KNN) approach tends to produce best document classification results. Although we have not tried this approach, it does not seem computationally viable to apply it to our training set of over thirty thousand documents, and we doubt that it would be the most appropriate technique for our multi-class categorisation problem. Unpublished experiments in which we trained Support Vector Machines (SVMs)

[1] `http://europa.eu.int/comm/food/fs/inspections/vi/reports/austria/vi_rep_oste_1074-1999_en.html`

with our data (one classifier per EUROVOC descriptor) produced rather bad results. However, first results in ongoing experiments on using SVM classifiers to eliminate wrong hits in the list of EUROVOC descriptors produced with the method described in Pouliquen et al. (2003a) are very promising. The method used is language-independent.It has been applied to all eleven official EU languages with very similar results, but the parameter settings have currently only been optimised for English and Spanish, and to a lesser degree for French. We hope to soon be able to map documents in some of the EU's new languages to EUROVOC.

The system was trained on 30231 texts and it was evaluated on a complementary test set of 590 representative, but randomly chosen texts. The descriptor assignment was evaluated against the annotation of two separate indexing professionals. The precision achieved for the eight highest-ranking descriptors assigned automatically (eight was the average number of descriptors assigned manually) was 67% (F-measure=65). This is 86% as good as the inter-annotator agreement of 78% (67/78=86%).

4 Calculation of cross-lingual document similarity

Each EUROVOC descriptor is identified by a numerical code and has one-to-one translations into the various languages. Texts can be represented as a vector consisting of the EUROVOC descriptor codes assigned to them and their assignment score. Two texts can then be compared with each other by calculating the cosine similarity between the two descriptor vectors. The higher the similarity value, the more similar the texts. The similarity measure for documents is the same, independent of the document languages. Table 2 shows the words (the associates) found in the news clusters in all five languages that triggered the EUROVOC descriptor POLITICAL COOPERATION. This descriptor was one of the most important descriptors out of many that helped to identify the related news clusters across the five languages.

The performanceof the CLDS was tested on a parallel corpus of 820 English-Spanish document pairs. The task was to search for the most similar Spanish document for each English document. If the translation was automatically identified as the most similar document, the experiment was considered successful, otherwise it was a failure. The experiments, which were described in detail in Pouliquen et al. (2003b), showed that the system successfully identifies the translation within the search space of 820 documents in over 90% of all cases. When also using document length as a feature, the performance goes up to 97%. Performance drops to 77% in a bilingual search space (searching for the most similar document among the Spanish *and* the English documents) because the average document similarity between the translation pairs is 0.83, i.e., significantly below 1. However, we were able to fully recover the performance by punishing

GERMAN	ENGLISH	SPANISH	FRENCH	ITALIAN
irak	nato	otan	irak	nesuna
g8	summit	irak	moyen	ruolo
gipfel	chirac	alianza	g8	intesa
nato	iraq	atlántica	orient	nato
rolle	united	cumbre	réformes	g8
island	debt	implicación	dette	rimandata
sea	middle	miembros	région	popolo
präsident	iraqi	funcionario	pays	iraq
us	g8	rusia	divergences	iracheno
einsatz	syria	chirac	dirigeants	incontro
initiative	east	iraquí	partenariat	georgia
schulden	democracy	polonia	sommet	accordo
staats	countries	francia	soutien	presidente
...	...	...	...	...

Table 2: *Lists of associates found in the news clusters of Figure 1 that triggered the EUROVOC descriptor POLITICAL COOPERATION*

the monolingual document similarity score with the multiplication factor 0.83. The CLDS results are thus clearly much better than the EUROVOC descriptor assignment results themselves, even though the CLDS is based on the automatically assigned descriptors. The reason presumably is that assignment consistency is more important than assignment accuracy.

5 Application to multilingual news analysis

We applied the CLDS calculation method to automatically establish links between news articles in the five languages English, German, French, Spanish and Italian of a large in-house collection of newspaper articles and news wires. The collection was compiled by the JRC's *Europe Media Monitor* system (EMM; Best et al. 2002), which gathers about 15000 news items per day (of which about 3000 written in English) in currently fifteen languages. The CLDS evaluation in Pouliquen et al. (2003b) shows that CLDS performance is best for the type of documents on which the system has been trained, and that it decreases with the degree in which the text corpus differs from the training corpus. We do not have access to a parallel corpus of news articles to carry out a quantitative test on this text genre, but an intuitive evaluation of descriptor assignment results to individual news articles showed that EUROVOC indexing for news items is much less successful. The reasons for this may either be that the news articles are too short to give enough lexical evidence on which to base the assignment, or the vocabulary used in news reporting differs too much from that of the parliamentary and legal training texts. To overcome this text genre problem, we made two changes to the original task of establishing cross-lingual links between news items: (1) We

additionally use named entities in the CLDS process and (2) we apply CLDS to *clusters*of news items rather than individual news feeds.
Using named entities (references in text to people, geographical places, organisations, etc.) was shown to be useful to improve the results of CLIR (Friburger & Maurel 2002). The relevance of named entities is particularly important for news analysis,as was shown by Clifton et al. (1999), whose (monolingual English) TopCat system identifies the major news topics of the day, based exclusively on named entities. We do not have access to general named entity recognition software, but we do have an in-house system to automatically identify and disambiguate references to geographical place names(Ignat et al. 2003, Pouliquen et al. 2004). In our current CLDS approach for news items, similarity based on geographical references contributes with a weight of one third, while CLDS based on EUROVOC descriptor assignment weighs two thirds.

keywords: **nato bush yawer summit** (18 articles from 10 newspapers)

10/06/2004 05:22 NEWScomAU: World leaders wrap up G8 summit

Gardian	Unity Masks Discord Between U.S., Europe
UsaToday	Greater role for NATO sought SEA ISLAND, Ga. – Seeking to build on rare harmony
CBSnews	G8 Summit: Smiles & Splits
Aljazeera-en	Chirac blocks Bush on NATO role in Iraq
GlasgowHerald	Chirac snubs wider NATO role in Iraq
CBSnews	Bush Not Counting On NATO Troops

...

Jacques Chirac[14], **Jiro Okuyama**[6], **Ghazi al-Yawer**[5], **Junichiro Koizumi**[6]
Recep Tayyip Erdogan[6], **Tony Blair**[5], **George W. Bush**[5], **Gerhard Schroeder**[5]

- 68% in Spanish Chirac condiciona la imlicación de la OTAN en Irak ...
- 51% in French Divergences au G8 malgré le soutien aux réformes au Moyen-Orient
- 68% in German Streit beim G8-Gipfel über Irak-Politik
- 64% in Italian G8, Bush-Chirac: nessuna intesa su ruolo Nato in Iraq

- 85% day - 1 Leaders call for NATO role
- 40% day - 4 Bush Looks Past Differences With France
- 41% day - 5 Bush Calls Terror 'Challenge of Our Time'

Figure 1: *Automatically identified English news cluster consisting of 18 articles. The system automatically produces a list of all articles and of person names mentioned in these articles. The hypertext links show the related clusters of Spanish, French, German and Italian news, as well as related news published and clustered over the previous days*

The clustering of news items according to the discussed event or subject is done using an agglomerative hierarchical method (Jain & Dubes 1988). We identify groups of news items by selecting the clusters that have a similarity of 50% or more in the clustering tree. We furthermore impose that the size of the subtree has to contain at least 0.6% of all articles in this language, and that the articles come from at least two different news sources. These thresholdswere chosen in an empirical refinement process because they group similar articles satisfactorily while doing well at distinguishing news clusters from other similar clusters.

Thirdly, they produce an average of about ten major news clusters per language and per day, which is more or less the number of news stories users in the European Commission requested for an automatically generated daily news overview. Links to the news clusters of the previous or following days are then established by calculating the similarity of the *clusters* of each day with each other. For each cluster, we also identify the centroid and identify the news article that is most similar to this centroid. Its title can then be used as the title for the cluster and users can read the chosen article to get informed about the subject, while links to the other articles of the cluster are available in case users want to read more about the event (see Figure 1). For each cluster, a mapis automatically generated to give an overview of the geographical place names mentioned in the whole cluster (using the methods described in Pouliquen et al. 2004).

Clustering the news of the day in five languages, identifying the geographical references, generating the maps and assigning EUROVOC descriptors currently takes about one hour on an average desktop PC. Establishing links between the languages and with the other days of the week is very fast and can be done online, if needed. There is no obvious quantitative way to evaluate the clustering and cross-lingual linking process, but from the application and user satisfaction points of view, the method works very well.

6 Future work

The results of the automatic clustering and the cross-lingual and temporal linking of news clusters are currently being incorporated with the EMM system (Best et al. 2002) in order to automatically generate daily news overviews for each day of the year in each of the languages covered. It is planned to extend the effort to all eleven official EU languages. We would also like to integrate further named entity recognition tools, because at least datesand names of people or organisations would contribute valuable clues for news clustering and even for cross-lingual linking.

The current clustering and cluster-linking system is focusing on the *major* newsof the day, i.e., those news that generated most news items. In a parallel effort, the JRC's EMM and Language Technology groups are working on *breaking news*detection. Breaking news are different from major news in that they are about new, *unexpected* events, such as a volcano eruption, a sudden flood or a terrorist attack. Breaking news are in principle identifiable because they should form a new cluster without any links to previous days. While detecting new clusters for a day is trivial, detecting breaking news for a smaller time window of an hour or less still represents an unsolved challenge. Establishing links between breaking news clusters in different languages will help to distinguish international breaking news from local or national events.

Acknowledgements. We would like to thank the European Parliament and the European

Commission's Office for Official Publications OPOCE for providing us with EUROVOC and the training material. We are grateful to the JRC's Web Technology group for their co-operation and for providing us with the valuable collection of multilingual news articles and news wires.

REFERENCES

Best, C., E. Van der Goot, M. de Paola, T. Garcia & D. Horby. 2002. Europe Media Monitor - EMM. JRC Technical Note No. I.02.88. JRC, Ispra, Italy.

Clifton, R., R. Cooley, J. Rennie. 1999. "TopCat: Data Mining for Topic Identification in a Text Corpus". *3rd Euro Conf. on Principles and Practice of Knowledge Discovery in Databases*, 174-183. Prague, Czech Republic.

Eurovoc. 1995. Thesaurus EUROVOC - vol. 2: Subject-Oriented Version. Ed. 3. English Language. Annex to the index of the Official Journal of the EC. Luxemburg. — `http://europa.eu.int/celex/EUROVOC` [Source checked in May 2004]

Friburger, N. & D. Maurel. 2002. "Textual Similarity Based on Proper Names". *Mathematical/Formal Methods in Information Retrieval Workshop* (*MFIR'02*) *at the 25th ACM SIGIR Conference*, 155-167. Tampere, Finland.

Ignat, C., B. Pouliquen, A. Ribeiro & R. Steinberger. 2003. "Extending an Information Extraction Tool Set to Central and Eastern European Languages". *Workshop on Information Extraction for Slavonic and other Central and Eastern European Languages*, 33-39. Borovets, Bulgaria.

Jain, A. & R. Dubes. 1988. *Algorithms for Clustering Data*. Englewood Cliffs, N.J.: Prentice-Hall.

Landauer, T. & M. Littman. 1991. "A Statistical Method for Language-Independent Representation of the Topical Content of Text Segments". *11th Conference 'Expert Systems and Their Applications'*, vol. 8, 77-85. Avignon, France.

Peters, C. 2003. "Cross Language Evaluation Forum". *Working Notes for the CLEF'03 Workshop*. Trondheim, Norway. — `http://clef.iei.pi.cnr.it`

Pouliquen, B., R. Steinberger & C. Ignat. 2003a. "Automatic Annotation of Multilingual Text Collections with a Conceptual Thesaurus". *EUROLAN workshop 'Ontologies & Information Extraction'*, 19-28. Bucharest, Romania.

Pouliquen, B., R. Steinberger & C. Ignat. 2003b. "Automatic Identification of Document Translations in Multilingual Document Collections". *Recent Advances in Natural Language Processing* (*RANLP-2003*), 401-408. Borovets, Bulgaria.

Pouliquen, B., R. Steinberger, C. Ignat & T. De Groeve. 2004. "Geographical Information Recognition and Visualisation in Texts Written in Various Languages". *19th Annual ACM Symposium on Applied Computing* (*SAC'2004*), *'Information Access and Retrieval' track* (*SAC-IAR*), vol. 2, 1051-1058. Nicosia, Cyprus.

Salton, G. & C. Buckley. 1988. "Term-Weighting Approaches in Automatic Text Retrieval". *Information Processing & Management* 24:5.513-523.

Sebastiani, F. 1999. "A Tutorial on Automated Text Categorisation". *Proceedings of the 1st Argentinean Symposium on Artificial Intelligence* (*ASAI-99*) ed. by A. Amandi & A. Zunino, 7-35. Buenos Aires, Argentina.

Verb Phrase Ellipsis Detection Using Machine Learning Techniques

LEIF ARDA NIELSEN
King's College London

Abstract

Although considerable work exists on the subject of ellipsis resolution, there has been very little empirical, corpus-based work on it. This paper describes work done on the detection of instances of Verb Phrase Ellipsis, using machine learning techniques. The goal of the system is to be robust, accurate and domain-independent.

1 Introduction

Ellipsis is a linguistic phenomenon that has received considerable attention, mostly focusing on its interpretation. An example of Verb Phrase Ellipsis (VPE)[1], which is detected by the presence of an auxiliary verb without a verb phrase, is seen in Example 1.

(1) *John read the paper before Bill did.*

Insight has been gained through work aimed at discerning the procedures and the level of language processing at which ellipsis resolution takes place. Such work has generally resulted in two views: syntactic (Fiengo & May 1994, Lappin & McCord 1990, Lappin 1993, Gregory & Lappin 1997) and semantic (Dalrymple et al. 1991, Kehler 1993, Shieber et al. 1996). Both views have their strengths and weaknesses, but they have so far not been validated using a corpus based, empirical approach, meaning that their actual performance is unknown. Furthermore, while these approaches take difficult cases into account, they do not deal with noisy or missing input, which is unavoidable in real NLP applications. They also do not allow for focusing on specific domains or applications, or different languages. It therefore becomes clear that a robust, trainable approach is needed.

Furthermore, they usually do not deal with the detection of elliptical sentences or the identification of the antecedent and elided clauses within them, but take them as given, concentrating instead on the resolution of ambiguous or difficult cases.

This paper describes the work done at the stage of the detection of elliptical verbs. After a heuristic baseline is built, a number of machine learning algorithms are used to achieve higher performance.

[1] We have chosen to concentrate on VP ellipsis due to the fact that it is far more common than other forms of ellipsis, but pseudo-gapping has also been included due to the similarity of its resolution to VPE (Lappin 1996).

2 Detection of elided VPEs

2.1 *Previous work*

The only empirical experiment done for this task to date, to our knowledge, is Hardt's (1997) algorithm for detecting VPE in the Penn Treebank. It achieves precision levels of 44% and recall of 53%, giving an F1 of 48% using a simple search technique, which relies on the annotation having identified empty expressions correctly. It should be noted that while Hardt's results are low, this is to be expected as his search in the Treebank is achieved by looking for a simple pattern: (VP (-NONE- *?*)). Such low performance can lead to incorrect conclusions being drawn through analysis, so it becomes clear that an initial stage with higher performance is necessary.

2.2 *Experimental method and data*

The British National Corpus (BNC) will be the corpus used for initial experiments. The gold standard was derived by marking the position in sentences where an elided verb occurs[2]. The performance of the methods is calculated using recall, precision and the F1-measure[3].

A range of sections of the BNC, containing around 370k words[4] with 712 samples of VPE was used as training data. The separate test data consists of around 74k words[5] with 215 samples of VPE. The sections chosen from the BNC are all written text, and consist of extracts from novels, autobiographies, scientific journals and plays. The average frequency of VPE occurrences for the whole data is about once every 480 words, or once every 32 sentences.

[2] Currently only one annotator is available, but this will be remedied as soon as possible.

[3] Precision and recall are defined as :

$$Recall = \frac{No(\text{correct ellipses found})}{No(\text{all ellipses in test})} \quad (1)$$

$$Precision = \frac{No(\text{correct ellipses found})}{No(\text{all ellipses found})} \quad (2)$$

The F1 provides a measure that combines these two at a 1/1 ratio.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

[4] Sections CS6, A2U, J25, FU6, H7F, HA3, A19, A0P, G1A, EWC, FNS, C8T

[5] Sections EDJ and FR3.

2.3 *Baseline approach*

A simple heuristic approach was developed to form a baseline. The method takes all auxiliaries as possible candidates and then eliminates them using local syntactic information in a very simple way. It searches forwards within a short range of words, and if it encounters any other verbs, adjectives, nouns, prepositions, pronouns or numbers, classifies the auxiliary as not elliptical. It also does a short backwards search for verbs. The forward search looks 7 words ahead and the backwards search 3. Both skip 'asides', which are taken to be snippets between commas without verbs in them, such as : "... papers do, however, show ...".

The algorithm was optimized on the development data, and achieves recall of 89.60% and precision of 42.14%, giving an F1 of 57.32%. On the test data, recall is 89.30%, precision 42.76%, and F1 57.82%.

2.4 *Transformation-based learning*

As the precision of the baseline method is not acceptable, we decided to investigate the use of machine learning techniques. Transformation based learning (Brill 1995) was chosen as it is very flexible and a powerful learning algorithm.

Generating the training samples is straightforward for this task. We trained the μ-TBL system[6] (Lager 1999) using the words and POS tags from BNC as our 'initial guess'. For the gold standard we replaced the tags of verbs which are elliptical with a new tag, 'VPE'.

2.4.1 *Generating the rule templates*

The learning algorithm needs to have the rule templates in which it can search specified. A combination of templates, with permutations of single or groups of tags in the 3 word neighbourhood were used. We would have liked to use larger contexts, and finer permutations, but the number of permutations gets too large for the learning algorithm, which runs out of memory.

The results seen in Table 1 are obtained, where the threshold is the value new rules need to satisfy in number of improvements, or the algorithm stops learning. Lower thresholds mean more rules are learned, but also increases the likelihood of spurious rules being learned, i.e. recall increases, but at a cost to precision.

Threshold	Recall	Precision	F1
5	50.93	79.56	62.11
3	62.15	75.56	68.20

Table 1: *Results for initial transformation based learning*

[6] Downloadable from `http://www.ling.gu.se/~lager/mutbl.html`

2.4.2 *POS grouping*

Despite the fact that the training data consists of 370k words, there are only around 700 elided verbs in it. The scarceness of the data limits the performance of the learner, so a form of smoothing which can be incorporated into the transformation based learning model is needed. To achieve this, auxiliaries were grouped into subcategories of 'VBX', 'VDX' and 'VHX', where 'VBX' generalises over 'VBB', 'VBD' etc. to cover all forms of the verb 'be'; 'VHX' generalises over the verb 'have' and 'VDX' over the verb 'do'.

The results of this grouping on performance are seen in table 2. It is seen that both precision and recall are increased, with F1 increasing by more than 5%. It may be noted that modifying the rules learned does not change the recall for this experiment, but this is a coincidence; while the numbers are the same, there are differences in the samples of ellipses found.

Threshold	Recall	Precision	F1
5	68.22	82.02	74.49
5	68.22	79.78	73.55
3	68.69	79.03	73.50
3	68.69	76.56	72.41

Table 2: *Results for partially grouped transformation based learning*

To further alleviate the data scarcity, we then grouped all auxiliaries to a single POS category, 'VPX'. The results of this grouping on performance are seen in Table 3. It is seen that both precision and recall are increased, with F1 increasing by more than 8%. These experiments suggest that for the task at hand, the initial POS tag distinctions are too fine grained, and the system benefits from the smoothing achieved by grouping.

Threshold	Recall	Precision	F1
5	69.63	85.14	76.61
3	71.03	82.61	76.38

Table 3: *Results for grouped transformation based learning*

The top 5 rules learned after this grouping are seen in Table 4. The first column shows the score of the rule, which is how many corrections it made to resemble the gold standard more. The second column is which tags it changes, and the third column what tag it changes them into. The last column indicates the conditions for the rule to be applied. The rules which are learned by the system are quite simple, such as '[He laughed], did/didn't he ?' (rule 1/2), '[He] did so' (rule 5).

Rank	Score	Change	to	if
1	150	VPX	VPE	tag[1]=XX0 and tag[2]=PNP and wd[-1]=,
2	130	VPX	VPE	tag[1]=PNP and tag[2]=PUN and wd[-1]=,
3	86	VPX	VPE	tag[1]=XX0 and tag[2]=PUN and wd[1]=nt
4	64	VPX	VPE	tag[-1]=PNP and wd[1]=.
5	36	VPX	VPE	tag[1]=AV0 and tag[2]=PUN and wd[1]=so

Table 4: *Top 5 rules learned by further grouped transformation based learning*

2.5 *Maximum entropy modelling*

Maximum entropy modelling uses features, which can be complex, to provide a statistical model of the observed data which has the highest possible entropy, such that no assumptions about the data are made. These models differ from the transformation based learning algorithm described in section 2.4 in that a probability is returned as opposed to a binary outcome, and do not produce easily readable rules like TBL. Ratnaparkhi (1998) makes a strong argument for the use of maximum entropy models, and demonstrates their use in a variety of NLP tasks. The OpenNLP Maximum Entropy package[7] was used for the experiments.

2.5.1 *Feature selection*

The training data for the algorithm consists of each verb in the corpus, its POS tag, the words and POS tags of its neighbourhood, and finally a true/false attribute to signify whether it is elliptical or not.

Experiments with different amounts of forward/backward context give the results seen in Table 5. The threshold for accepting the results for a potential VPE were set to 0.2, or 20%; this value was determined by examining results.

Context size	Recall	Precision	F1
2	76.16	53.79	63.05
3	72.43	61.26	66.38
4	64.48	60.00	62.16

Table 5: *Results for maximum entropy learning*

The results show that for larger contexts the algorithm runs into problems. This is due to the fact that the contexts do not allow for the kind of generalization available to transformation based learning.

[7] Downloadable from `https://sourceforge.net/projects/maxent/`

2.5.2 *Thresholding*

Setting the forward/backward context size to 3, experiments are run to determine the correct setting for the threshold. This value is used to determine at what level of confidence from the model a verb should be considered a VPE.

Context size	Recall	Precision	F1
0.10	78.50	47.86	59.46
0.15	76.16	54.88	63.79
0.20	72.43	61.26	66.38
0.25	68.22	65.17	66.66
0.30	63.08	67.16	65.06
0.35	59.34	70.16	64.30
0.40	57.00	71.76	63.54
0.45	52.80	73.85	61.58
0.50	49.53	74.64	59.55

Table 6: *Effects of thresholding on maximum entropy learning*

With higher thresholds, recall decreases as expected, while precision increases. F1 peaks at 0.25, which is close to the initial guess of 0.2. The fact that this value is so low is expected to be due to the size of the corpus. For subsequent experiments, a threshold of 0.2 will be retained, for comparison purposes, and because its results are very close to those of 0.25.

2.5.3 *POS grouping*

Using the same principles for smoothing introduced in section 2.4.2, the effects of category grouping are investigated, with the results seen in Table 7 for fully grouped data.

Context size	Recall	Precision	F1
2	77.57	55.14	64.46
3	76.16	65.72	70.56
4	67.28	64.00	65.60

Table 7: *Results for grouped maximum entropy*

It is interesting to note that the effect of grouping for maximum entropy is less than that for transformation based learning; 4% compared to 8%. Furthermore, seen from the perspective of error reduction, grouping only gives a 15% reduction for maximum entropy, while transformation based learning gets a 25% error reduction.

2.6 *Decision tree learning*

Decision trees, such as ID3 (Quinlan 1986) and C4.5 (Quinlan 1993), contain internal nodes, which perform tests on the data, and following the result of these test through to the end leaf, gives the probability distribution associated with it. These methods have the advantage that they automatically discard any features that are not necessary, can grow more complicated tests from the tests (features) given, and that the resulting trees are usually human readable.

2.6.1 *Data decimation*

The C4.5 algorithm works in two steps: first, a large tree is constructed that creates simple rules for every example found, and second, more generalized rules are extracted from this tree. Running both parts of the algorithm, C4.5 (statistically correctly) deduces that the best rule to learn is that there is no need to recognize VPEs, and everything is non-elliptical, as this results in only a 1.4% error overall. This fits with C4.5's design, which is to not overfit data, and produce few, general rules.

It was only possible to get results from the algorithm by removing non-elided samples from the training corpus, to artificially weigh elided samples, and only running the first part of C4.5. Discarding every 20th sample, with full grouping, precision of 79.39% and recall of 60.93% is obtained, giving an F1 of 68.94%.

2.7 *Memory-based learning*

Memory based learning is a descendant of the classical *k*-Nearest Neighbour approach to classification. It places all training instances in memory and classifies test instances by extrapolating a class from the most similar instances. It has been used for a variety of NLP tasks, and the technique is meant to be useful for sparse data, as its feature weighing produces smoothing effects. We used TiMBL (Daelemans et al. 2002), training it with the same data used for the maximum entropy and C4.5 experiments.

Table 8 shows the results obtained. Again, a context size of 3 gives best results.

Context size	Recall	Precision	F1
2	71.49	69.23	70.34
3	73.83	72.14	72.97
4	72.42	69.50	70.93

Table 8: *Results for* MBL

2.7.1 *POS grouping*

Using the same principles for smoothing introduced in section 2.4.2, the effects of category grouping are investigated, with the results seen in Table 9, gives a 2% increase in F1 over non-grouped data, for a context size of 3.

Context size	Recall	Precision	F1
2	74.29	68.24	71.14
3	76.16	73.75	74.94
4	73.83	70.53	72.14

Table 9: *Results for grouped* MBL

It is interesting that while MBL achieves higher results for non-grouped data than the other algorithms, it is also the one to benefit the least from grouping, suggesting that grouping is more useful for generalizing rules than it is for pattern matching.

2.8 *Combining algorithms*

As the different algorithms tested produce results with different characteristics, their results can be seen as refined features. The baseline algorithm, for example, has high recall but low precision, while the opposite holds true for TBL. Experiments were conducted to see if using their results as features, alongside the words and their POS, results in improved performance.

Alg.	3 Context			5 Context		
	Rec	Pre	F1	Rec	Pre	F1
MaxE	76.16	65.72	70.56	64.95	67.47	66.19
+ B	82.71	71.08	76.45	78.03	73.24	75.56
+ T	71.02	74.50	72.72	69.15	80.87	74.55
+ BT	76.16	75.46	75.81	75.23	78.53	76.84
C4.5	60.00	79.14	68.25	60.93	79.39	68.94
+ B	41.12	80.00	54.32	41.12	80.00	54.32
+ T	66.35	86.06	74.93	66.35	86.06	74.93
+ BT	66.35	86.06	74.93	66.35	86.06	74.93
MBL	76.16	73.75	74.94	73.36	69.77	71.52
+ B	77.57	77.20	77.38	76.63	75.57	76.10
+ T	75.70	77.51	76.59	74.29	75.00	74.64
+ BT	76.16	78.74	77.43	75.23	76.30	75.76

Table 10: *Combining algorithms*

The results in Table 10 show that this approach does not produce significant gains, with the highest F1 produced at 77.4% by MBL, using just the baseline or the baseline plus TBL.

It is interesting to note that the contribution of baseline features and TBL features are about the same for Maxent and MBL, with baseline features giving better results. Maxent and MBL produce balanced precision and recall scores, despite the differing natures of the added data. This suggests that they don't generalize, even to TBL which is quite precise, and learn rules to augment it. On the other hand, when given TBL data, C4.5 will default to it. When given just baseline data though, it learns a number of rules it trusts more.

3 Conclusion and future work

Experiments utilizing four different machine learning algorithms have been conducted on the task of detecting VPE instances. Their performances are summarized in Table 11.

Algorithm	Recall	Precision	F1
Baseline	89.30	42.76	57.82
TBL	69.63	85.14	76.61
MaxEnt	76.16	65.72	70.56
C4.5	60.93	79.39	68.94
MBL	76.16	73.75	74.94

Table 11: *Comparison of algorithms*

It must be noted that the results achieved are limited by the training data size, which had to be kept to 370k words due to problems encountered with the μ-TBL learning algorithm. This has also meant that for the TBL experiments, as wide a range of contexts as was expected could not be used. The training data size could have been increased for the other algorithms, but was not for comparison purposes.

Augmenting MBL with the baseline results produces a 77.4% F1. This gives a 19.6% increase over our baseline, and a 29.4% increase over Hardt's results, although they are not directly comparable due to the different corpora used.

The results so far are encouraging, and show that the approach taken is capable of producing a robust and accurate system. Future work will concentrate on using parsed data to investigate the effect of the extra information on performance, and increasing the training dataset.

REFERENCES

Brill, Eric. 1995. "Transformation-based Error-driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging". *Computational Linguistics* 21:4.543-565.

Daelemans, Walter, Jakub Zavrel, Ko van der Sloot & Antal van den Bosch. 2002. "Tilburg Memory Based Learner, version 4.3, Reference guide". — `http://ilk.kub.nl/downloads/pub/papers/ilk0210.ps.gz` [Source checked in May 2004]

Dalrymple, Mary, Stuart M. Shieber & Fernando Pereira. 1991. "Ellipsis and Higher-order Unification". *Linguistics and Philosophy* 14:399-452.

Fiengo, Robert & Robert May. 1994. *Indices and Identity*. Cambridge, Mass.: MIT Press.

Gregory, Howard & Shalom Lappin. 1997. "A Computational Model of Ellipsis Resolution". *Formal Grammar Conference*, Aix-en-Provence, France.

Hardt, Daniel. 1997. "An Empirical Approach to Ellipsis". *Computational Linguistics* 23:4.525-541.

Kehler, Andrew. 1993. "A Discourse Copying Algorithm for Ellipsis and Anaphora Resolution". *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics* (*EACL-93*), 203-212. Utrecht, The Netherlands.

Lager, Torbjorn. 1999. "The μ-TBL System: Logic Programming Tools for Transformation-Based Learning". *Third International Workshop on Computational Language Learning* (*CoNLL'99*). — `http://ilk.kub.nl/downloads/pub/papers/ilk0210.ps.gz` [Source checked in May 2004]

Lappin, Shalom. 1993. "The Syntactic Basis of Ellipsis Resolution". *Proceedings of the Stuttgart Ellipsis Workshop, Arbeitspapiere des Sonderforschungsbereichs 340, Bericht Nr. 29-1992* ed. by S.Berman & A.Hestvik. Stuttgart: Univ. of Stuttgart.

Lappin, Shalom. 1996. "The Interpretation of Ellipsis". *The Handbook of Contemporary Semantic Theory* ed. by Shalom Lappin, 145-175. Oxford: Blackwell.

Lappin, Shalom & Michael McCord. 1990. "Anaphora Resolution in Slot Grammar". *Computational Linguistics* 16:4.197-212.

Quinlan, J. R. 1986. "Induction of Decision Trees". *Readings in Machine Learning* 1:1.81-106.

Quinlan, J.R. 1993. *C4.5: Programs for Machine Learning*. San Mateo, Calif.: Morgan Kaufmann.

Ratnaparkhi, Adwait. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. dissertation, Univ. of Pennsylvania. Philadelphia.

Shieber, Stuart M., Fernando Pereira & Mary Dalrymple. 1996. "Interactions of Scope and Ellipsis". *Linguistics and Philosophy* 19:5.527-552.

HPSG-based Annotation Scheme for Corpora Development and Parsing Evaluation

KIRIL IV. SIMOV
Bulgarian Academy of Sciences

Abstract

This paper proposes a formal framework for development and exploitation of a corpus, based on the HPSG linguistic theory. The formal representation of the annotation scheme facilitates the annotation process and ensures the quality of the corpus and its usage in different application scenarios. Also, evaluation over HPSG annotation scheme is discussed.

1 Introduction

This paper proposes a strategy for the construction of a corpus, based on the HPSG (Head-driven Phrase Structure Grammar) linguistic theory (see Pollard & Sag 1994). The annotation scheme for the corpus is formally defined in a formalism for HPSG and reflects the developments in the theory. Thus our approach tries to incorporate both - the language theory and the underlying formal assumptions. It has the following important advantages: (1) By imposing constraints over the HPSG-derived annotation scheme, the annotation process becomes more efficient; (2) It supports the definition of validation theories, which encode more consistently the otherwise informal annotation guidelines; (3) When the annotation scheme is changed at some later stage of the corpus development, the previously annotated sentences can be reclassified with respect to the new scheme on the basis of: (a) the information that has already been encoded, and (b) the minimal human intervention; (4) The formalism allows for different levels of abstraction over the data in the corpus. This can be very useful for the application possibilities of the corpus. For example, different learning mechanisms might rely upon different types of information. (5) Also the inference mechanisms can be used for evaluating parsers with respect to such a corpus.

The work reported here has been developed within the BulTreeBank project (Simov, Popova & Osenova 2002). The main goal of the project being the construction of an HPSG-based treebank for Bulgarian. This goal presupposes the choice of the linguistic theory and its adequate formalization. In our case it is the HPSG theory and the SRL grammar formalism that we rely upon. The choice is motivated by the fact that HPSG and SRL meet the requirements for a consistent representation of the linguistic knowledge within the treebank. Of course, there exist other formalisms for the HPSG theory, but the comparison with them is beyond the scope of this paper. Note that the ideas presented here can be worked out for other grammar formalisms as well as for other grammar theories.

We would like to discuss briefly some often raised questions with respect to the development of a treebank, namely, the role of the linguistic theory, and the relation between a certain grammar and the treebank. Concerning the role of the linguistic theory, we believe that the notion of 'theory independency' is impossible in case of detailed linguistic description, if at all. In our view, the annotation scheme of each treebank always involves some linguistic theory, especially when taking specific decisions on the representation of the linguistic facts and their interrelation. The connection between the grammar and treebank development is bidirectional. On the one hand, a recent survey on treebanks (Abeillé 2003) shows that most of the treebanks are grammar-based in the following sense: they use a pre-defined grammar for the production of all the possible sentence analyses, which later are manually corrected. The main advantage of such a treebank is the additional knowledge, entered by the annotator in the post-parsing phase. On the other hand, the constructed treebanks can be used for grammar extraction and specialization. How a treebank of this kind can be exploited for such purposes, is described elsewhere (Simov 2002, Simov et al. 2002).

The structure of the paper is as follows: Section 2 describes the formalism that is employed for the representation of the HPSG grammar, the corpus and the annotation scheme. Section 3 demonstrates how this formalism can be used for facilitating the annotation process. Section 4 focuses on the reclassification algorithm. Section 5 presents the evaluation process. Section 6 discusses related works. The last section concludes the paper.

2 Formalism for HPSG

In this section we present shortly the logical formalism (SRL) for HPSG which we use in our work. For full description see (King 1989). A normal form for a finite SRL theory is defined as a set of feature graphs (see (King & Simov 1998)). (Simov 2002) shows that this normal form is suitable for the representation of an HPSG corpus and an HPSG grammar.

$\Sigma = \langle \mathcal{S}, \mathcal{F}, \mathcal{A} \rangle$ is a finite **SRL signature** iff $\mathcal{S}$ is a finite set of *species*, $\mathcal{F}$ is a set of *features*, and $\mathcal{A} : \mathcal{S} \times \mathcal{F} \rightarrow Pow(\mathcal{S})$ is an *appropriateness function*. τ is a **term** iff τ is a member of the smallest set $\mathcal{T}$, such that (1) $: \in \mathcal{T}$, and (2) for each $\phi \in \mathcal{F}$ and each $\tau \in \mathcal{T}$, $\tau\phi \in \mathcal{T}$. δ is a **description** iff δ is a member of the smallest set $\mathcal{D}$, such that (1) for each $\sigma \in \mathcal{S}$ and for each $\tau \in \mathcal{T}$, $\tau \sim \sigma \in \mathcal{D}$, (2) for each $\tau_1 \in \mathcal{T}$ and $\tau_2 \in \mathcal{T}$, $\tau_1 \approx \tau_2 \in \mathcal{D}$ and $\tau_1 \not\approx \tau_2 \in \mathcal{D}$, (3) for each $\delta \in \mathcal{D}$, $\neg\delta \in \mathcal{D}$, (4) for each $\delta_1 \in \mathcal{D}$ and $\delta_2 \in \mathcal{D}$, $[\delta_1 \wedge \delta_2] \in \mathcal{D}$, $[\delta_1 \vee \delta_2] \in \mathcal{D}$, and $[\delta_1 \rightarrow \delta_2] \in \mathcal{D}$. Each subset $\theta \subseteq \mathcal{D}$ is an **SRL theory**.

An HPSG grammar $\Gamma = \langle \Sigma, \theta \rangle$ in SRL consists of: (1) a signature Σ, which gives the ontology of entities that exist in the universe and the appropriateness conditions on them, and (2) a theory θ, which gives the restrictions upon these entities. We represent grammars and corresponding sentence analyses in a normal form based on feature graphs.

Let $\Sigma = \langle \mathcal{S}, \mathcal{F}, \mathcal{A} \rangle$ be a finite signature. A **feature graph** with respect to Σ is a directed, connected and rooted graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{V}, \rho, \mathcal{S} \rangle$, such that: (1) $\mathcal{N}$ is a set of **nodes**, (2) $\mathcal{V} : \mathcal{N} \times \mathcal{F} \rightarrow \mathcal{N}$ is a partial **arc function**, (3) ρ is a **root node**, (4) $\mathcal{S} : \mathcal{N} \rightarrow \mathcal{S}$ is a total **species assignment function**, and (5) for each $\nu_1, \nu_2 \in \mathcal{N}$ and each $\phi \in \mathcal{F}$, such that $\mathcal{V}\langle \nu_1, \phi \rangle \downarrow$ and $\mathcal{V}\langle \nu_1, \phi \rangle = \nu_2$, then $\mathcal{S}\langle \nu_2 \rangle \in \mathcal{A}\langle \mathcal{S}\langle \nu_1 \rangle, \phi \rangle$. We say that the feature graph $\mathcal{G}$ is **finite** if and only if the set of nodes is finite. A feature graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{V}, \rho, \mathcal{S} \rangle$, such that for each node $\nu \in \mathcal{N}$ and each feature $\phi \in \mathcal{F}$ if $\mathcal{A}\langle \mathcal{S}\langle \nu \rangle, \phi \rangle \downarrow$ then $\mathcal{V}\langle \nu, \phi \rangle \downarrow$ is called a **complete feature graph**. For each two graphs $\mathcal{G}_1 = \langle \mathcal{N}_1, \mathcal{V}_1, \rho_1, \mathcal{S}_1 \rangle$ and $\mathcal{G}_2 = \langle \mathcal{N}_2, \mathcal{V}_2, \rho_2, \mathcal{S}_2 \rangle$ we say that graph $\mathcal{G}_1$ **subsumes** graph $\mathcal{G}_2$ ($\mathcal{G}_2 \sqsubseteq \mathcal{G}_1$) iff there is an *isomorphism* $\gamma : \mathcal{N}_1 \rightarrow \mathcal{N}_2'$, $\mathcal{N}_2' \subseteq \mathcal{N}_2$, such that (1) $\gamma(\rho_1) = \rho_2$, (2) for each $\nu, \nu' \in \mathcal{N}_1$ and each feature ϕ, $\mathcal{V}_1\langle \nu, \phi \rangle = \nu'$ iff $\mathcal{V}_2\langle \gamma(\nu), \phi \rangle = \gamma(\nu')$, and (3) for each $\nu \in \mathcal{N}_1$, $\mathcal{S}_1\langle \nu \rangle = \mathcal{S}_2\langle \gamma(\nu) \rangle$. For each two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ if $\mathcal{G}_2 \sqsubseteq \mathcal{G}_1$ and $\mathcal{G}_1 \sqsubseteq \mathcal{G}_2$ we say that $\mathcal{G}_1$ and $\mathcal{G}_2$ are **equivalent.**

For finite feature graphs, we could define a translation into SRL descriptions using the correspondences between paths in the graph and terms. Thus we can interpret each finite feature graph as a description in SRL. Using the set of all finite feature graphs that subsume a given infinite feature graph, we can also define the interpretation of each infinite feature graph. Thus we can speak about satisfiable graphs. For them there exists an interpretation in which they denote a non-empty set of objects. Moreover, we can define a correspondence between the finite SRL theories and the sets of feature graphs. Thus we can represent each finite theory as a set of graphs, called *a graph theory representation*. This representation has the following important properties: (1) each graph $\mathcal{G}$ in the set of graphs is satisfiable (for some interpretation the graph $\mathcal{G}$ denotes some objects in the interpretation), and (2) each two graphs $\mathcal{G}_1$, $\mathcal{G}_2$ in the set have disjoint denotations (for each interpretation there is no object in the interpretation that is denoted by the two graphs). The mentioned above properties allow for classification of linguistic objects with respect to the classes defined by the graphs.

We use feature graphs and algorithms for their processing in the tasks connected to the creation and usage of an HPSG corpus. Next (Simov 2002) we assume that an annotation scheme over the HPSG sort hierarchy can be considered a grammar. The feature graphs of such an annotation scheme will be constrained by the lexicon, which is available to the annotators; by the principles, which are stated as a theory; and by the input sentences. As a result, all the constraints that follow logically from the above sources of information can be exploited during the annotation process.

3 Corpus annotation

The corpus annotation within this framework is based on the idea of parse selection from a number of automatically constructed sentence parses. The parses

are constructed by the inference mechanism using the graph representation of the annotation scheme and graph encoding of the sentence. This approach to corpora development is well known as *grammar-based corpus annotation*. See (Dipper 2000) for an example among others. Our approach differs in the formal mechanisms that are incorporated within the implementation. The assumptions, which the annotation process is based on, are: (1) The annotation scheme is defined as a set of graphs. Thus each sentence annotation has to be consistent with respect to the logical properties of the annotation scheme. Nevertheless, the annotator is not constrained too much, because the annotation scheme is still general and therefore, it will overgenerate massively. (2) The annotator cannot simultaneously observe all the parses, generated by the annotation scheme. He/she has to make a choice relying only on the partial information as a prompt for the sentence true analysis. Hence, annotators' work is incremental. (3) The information, added by the user during the annotation process, is propagated further. The propagation reduces the number of the possible choices in other places of the sentence analysis. Thus, the overall annotation process is organized as follows:

(1) First, the selected sentence is processed partially. This processing is compatible with the HPSG sort hierarchy and comprises: morphological analysis, disambiguation and non-recursive partial parsing. As a result, the complexity of the following steps is reduced. Note that the sentences receive a unique partial analysis.

(2) The result from the previous step is encoded as feature graph and it is further processed by an HPSG processor with the help of the described annotation scheme. The result is a set of complete feature graphs.

(3) The selection of the correct analysis is considered *a classification* of the partial description of the true sentence analysis with respect to the set of complete feature graphs, produced in the previous step. The classification starts with the information common for all complete graphs. This information contains all the partial analyses from the first step, because the HPSG processor operates monotonically and thus, it cannot delete information. On the basis of the differences between the complete graphs *an index* over them is created. This index supports the propagation of the information, added by the annotator. When the user adds enough information, the partial analysis can be extended to exactly one of the complete graphs. If the sentence allows more than one analysis, the annotator has to classify it more than once.

Next, we present the index and describe its contribution to the process of classification of the partial sentence analysis with respect to all sentence analyses. The idea is that the annotator states the new information about the analysis as elementary descriptions of the relevant graph. The elementary descriptions are of the following kinds: $\pi \sim \sigma$ (the path π is defined in the graph and the species of the end node is σ), $\pi \not\sim \sigma$ (the path π is defined in the graph and the species of the end node is not σ), $\pi_1 \approx \pi_2$ (the paths π_1 and π_2 are defined in

the graph and they share their end nodes), and $\pi_1 \not\approx \pi_2$ (the paths π_1 and π_2 are defined in the graph and they have different end nodes).

Here are some examples of elementary descriptions that the user can supply: "the phrase is of type *head-complement*", "the verbal adjunct is not a secondary predication", "the unexpressed subjects of two relative clauses are the same". If, for example, the sentence also contains reflexive pronouns, bound to the unexpressed subject in one of the relative clauses, the last claim will automatically add a binding link from this pronoun to the unexpressed subject of the other relative clause.

In (King & Simov 1998) we have shown that for a set of graphs, representing a theory, there is a set of elementary descriptions, such that each description in the set discriminates over the set of graphs. Thus, it is true for at least one graph and it is false for at least one graph. Using the last properties one can construct an index over the set of graphs. The index is a tree, such that the nodes of the tree are marked with elementary descriptions and the edges of the tree are marked with truth values: *true* or *false*. And the descriptions are chosen in such a way that each path from the root of the tree to some of the leaves of the tree determines exactly one graph in the initial set of graphs. The descriptions, presented in the index, can be chosen on the basis of the graphs.

In order to use such indices for facilitating the annotation process, we encode all possible indices over the complete graphs, returned by the HPSG processor. This work is being done incrementally over the differences of the graphs and thus the indices share some of their parts. The index is not a tree in this case, but rather a forest. This step is necessary for annotators' convenience, because it is not clear at the beginning, which information will be easy to be provided manually.

It can be proved that for a finite set of graphs there exists a finite index. Stating one of the elementary descriptions in the index, the annotator always reduces the number of the graphs that are presented by this description. Providing several descriptions, the annotator arrives at exactly one graph from the set. Thus, the classification is performed in the following way:

(1) At the beginning all the nodes in the index are available to be chosen and the annotator has the possibility to state any of the elementary descriptions in the index.

(2) The annotator decides on an elementary description about the sentence from the set of the allowed descriptions.

(3) The description is found in the index and the operation reduces the number of the possible graphs. It also means that some of the elementary descriptions in the index are not eligible any more, because they will contradict the selected description.

(4) If the set of the possible graphs is a singleton (has only one member), then this graph is a result from the classification. If the set contains more than

one graph, then the algorithm goes to point 2 and offers the annotator to make a new choice of an allowed elementary description.

The chosen graph is in fact the analysis of the sentence. It is important to say that this algorithm of classification works not only over a set of complete graphs, but also over graph representations of finite SRL theories.

An additional facility for the annotator is the possibility for him/her to provide larger descriptions at one step. Such descriptions represent the linguistically motivated characteristics of the sentence. Larger descriptions can be considered macros. For example, macroses are the constituent labels like VPS for verbal head-subject phrase, NPA for noun head-adjunct phrase etc.

As a speed measure of the annotation we consider all the necessary selections made by the annotator in his/her steps to the complete analysis. The number of the selections are in the worst case equal to the number of all analyses, produced by the HPSG grammar. This can happen when the annotator rules out exactly one analysis per choice. The average number of selections is a logarithm from the number of the analyses. An important advantage of this selection-analysis-approach is that the annotator works locally. Thus the number of parameters necessary to be considered simultaneously is minimized.

4 Reclassification

The need for a reclassification of already classified linguistic objects arises in connection with the following problems and tasks: (1) Changes in the target linguistic description of the elements in the corpus; (2) New tasks, for which the corpus might be adjusted; (3) New developments in the linguistic theory; (4) Misleading decisions, taken during the design phase of the corpus development. In each of these cases, the development of a new annotation scheme is necessary. The problems concerning such a step are well known: What about the corpus built up to now? How to use it in the new circumstances and at minimal costs? Here we offer an algorithm for reclassification within our formalism for HPSG.

There are two possible scenarios for the application of the reclassification to an already created corpus: (1) The first holds when the changes in the annotation scheme are relatively small. For instance, addition of new features, new sorts or new principles to the initial HPSG grammar; (2) In the second case there is a substantial change in the annotation scheme. For example, complete substitution of the sort hierarchy parts with new ones. Of course, there are not clear boundaries between the two kinds of changes.

Let Σ_{old} and Σ_{new} be two signatures and let A_{old} be the annotation scheme constructed on the basis of Σ_{old} and A_{new} be the annotation scheme constructed on the basis of Σ_{new}. The idea of reclassification is based on the notion of the *correspondence rules* between descriptions with respect to the old and to the new

annotation schemes. The general format of these rules is $\delta_{old} \Rightarrow \delta_{new}$ where the δ_{old} is a description with respect to Σ_{old} and δ_{new} is a description with respect to Σ_{new}. The meaning of such rules is: for each model $\mathcal{I}_{old}$ of A_{old}, such that the description δ_{old} is satisfiable in it, there exists a model $\mathcal{I}_{new}$ of A_{new}, such that the description δ_{new} is satisfiable in it. Thus we consider the correspondence rules as rules for transferring knowledge between the two annotation schemes.

Then the algorithm for reclassification works in the following way:

(1) Let CR is a set of correspondence rules between the two signatures Σ_{old} and Σ_{new}.
(2) Let $\mathcal{G}_{old}$ be a graph with respect to A_{old} for the sentence S. Let $\{\mathcal{G}^1_{new}, \ldots, \mathcal{G}^n_{new}\}$ are the candidate analyses for the sentence S with respect to the new scheme A_{new}.
(3) The algorithm constructs the set ED_{old} of all descriptions δ_{old}, such that there exists a correspondence rule $\delta_{old} \Rightarrow \delta_{new} \in CR$ and $\mathcal{G}_{old}$ is in the denotation of δ_{old} for each interpretation of A_{old} that satisfies $\mathcal{G}_{old}$. Thus ED_{old} contains all the descriptions on the left side of the correspondence rules that are true for the graph.
(4) Then the algorithm constructs the set ED_{new} of descriptions δ_{new}, such that there exists a correspondence rule $\delta_{old} \Rightarrow \delta_{new} \in CR$ and $\delta_{old} \in ED_{old}$. We consider the set ED_{new} to be the transferred knowledge from the old annotation of the sentence S to the new annotation.
(5) Then the algorithm uses the set ED_{new} and the index for the new potential analyses for the sentence S in order to find the minimal number of graphs from the set $\{\mathcal{G}^1_{new}, \ldots, \mathcal{G}^n_{new}\}$, which satisfies all the descriptions in ED_{new}.

The result of this algorithm is a set of graphs. If the set is empty, it means that the transferred knowledge is in contradiction with the new annotation scheme and cannot be really used. In this case the developers of the corpus have to reconsider the correspondence rules. If the set is a singleton, then it equals the analysis of the sentence with respect to the new annotation scheme. If the set contains more than one element, then the old analysis does not contain enough information for a unique classification of the sentence with respect to the new annotation scheme and some human intervention will be necessary. In fact we expect the last point to be the majority of the cases. Nevertheless the reclassification process will be helpful in this case also because it will reduce the number of the candidate analyses.

The two scenarios mentioned above differ from each other mainly on the basis of the complexity of the correspondence rules. In the first case one can state that each old description that is eligible with respect to the new scheme is mapped on itself. The second case will require more complicated rules.

5 Evaluation over an HPSG annotation scheme

A corpus, which is constructed with respect to this formalism, offers various opportunities for evaluation of parsing systems. Firstly, HPSG as a theory describes the linguistic objects by using both mechanisms for the representation of the syntactic information: constituent structure and head-dependent structure. Hence, one could rely on the most of the current evaluation metrics like: PARSEVAL precision and recall over bracketing and the mean number of overlapping brackets (Harrison et al. 1991), on evaluations focusing on grammatical relations as in (Carroll et al. 2003), or on dependency relations in (Kübler & Hinrichs 2001) and (Lin 2003). Additionally, such a corpus provides mechanisms for mixed evaluation schemes where both of these inventories can be used.

Another advantage of such a corpus is the high granularity of the information presented in it. This is a pre-requisite for the definition of different levels of degree where the evaluation process can take place. Note that it supports multi-level evaluation processes rather than mono-level ones. For example, one can work on the level of bracketing, but also she/he can view the constituents types as defined by the grammatical features of the head. Thus the two popular evaluation approaches can be easily implemented over the same corpus, i.e., either the bracketing precision and recall and bracketing overlap measures, or the grammar relations measures. One can even combine them in one measure parameter specific for certain evaluation requirements.

In order to achieve this one has to define a new annotation scheme, which reflects the evaluation task. Then it is necessary an appropriate set of correspondence rules to be defined. Afterwards the corpus is reclassified with respect to the new annotation scheme. In most cases this process will be a simplification of the information that is already in the corpus, and human intervention would not be necessary. For instance, one can keep only the information about head-dependents and delete all the information about the constituent structure. One important point here is that the deletion of information is not exactly transformation of the graphs that already exist in the corpus. In fact, this is a construction of a new corpus using the information stored in the old one. There is no need the graphs in the new corpus to be isomorphic to subgraphs in the old one. As a very simple example we can consider the transformation of a deep adjunct attachment (one at a time) into a flat adjunct attachment (all at once).

Another possibility is the context dependent evaluation. Generally, this means to mix several evaluation approaches depending on the linguistic information in the sentence analyses. For example, consider the case when the evaluation aims at determining the right argument recognition for the verbs, but not for the prepositions. Then one can require all NPs with attached to them PPs to be transformed into some flat structured NPs, and keep the PPs only when they are arguments of the verbs.

6 Related work and discussion

There are several existing works related to ours. Using graphs for representation of corpora data is presented in a number of papers of Steven Bird and co-workers (see (Cotton & Bird 2002) for applying their format to treebanks). The main difference between their approach and our work is that their graphs are defined purely in a graph-theoretical manner with some additions related to corpus practice. In our view some logical formalism for annotation graphs is necessary in order to ensure the consistency of the represented linguistic information and the result from the operations over them. For comparison, our feature graphs are directly related to well established logical formalism which ensures the necessary functionality for their manipulation. Also the expressive power of feature graphs seems to be greater than the one of annotation graphs.

Another related work is: Redwood HPSG treebank of English (Oepen et al. 2002). The creation of this treebank uses decision trees to support the annotators in the selection of the right HPSG analyses for the sentences. This approach is very close to our classification based on feature graphs. Another important characteristics of their treebank is its dynamic nature. This generally is concerned with changes and new developments in the underling linguistic theory. The problem is the following: when the theory changes the treebank becomes out of date. In order to support easily the updates of the already represented information one needs a mechanism for reusing of old analyses with small amount of work. The people working on Redwood treebank achieve this again by the means of decision trees. We can use reclassification for the same purpose.

The reclassification is also related to the approach described in (Kinyon & Rambow 2003) which is used for transformation of treebanks from one linguistic theory to another. Again the difference with our approach is that we offer these operations to be done on the basis of a logical formalism, and the consistency of the result is guaranteed when the original information is consistent.

7 Conclusions

In this paper we presented a formal framework for development of a corpus based on HPSG linguistic theory. There are several advantages of such a formal framework: (1) Uniformity of the annotation with respect to an HPSG grammar; (2) Classification algorithm for facilitating the annotation process; (3) Potential for reclassification which can be helpful during the development of the corpus and during its exploitation.

One interesting side of such usage is for parser evaluation. Firstly, HPSG as theory offers simultaneously representation of the constituent structure and the dependency relations. Secondly, the reclassification of the corpus can be context sensitive and this allows for different kinds of evaluation for different constructions.

REFERENCES

Abeillé Anne, ed. 2003. *Treebanks. Building and Using Parsed Corpora.* Dordrecht: Kluwer Academic.

Carroll, John, Guido Minnen & Ted Briscoe. 2003. "Parser Evaluation Using a Grammatical Relation Annotation Scheme". *Treebanks. Building and Using Parsed Corpora* ed. by Anne Abeillé, 299-316. Dordrecht: Kluwer Academic.

Cotton, Scott & Steven Bird. 2002. "An Integrated Framework for Treebanks and Multilayer Annotations". *Proceedings of the International Conference on Language Resources and Evaluation* (*LREC'02*), 1670-1677. Canary Islands, Spain.

Dipper, Stefanie. 2000. "Grammar-based Corpus Annotation". *Proceedings of the Workshop on Linguistically Interpreted Corpora*, 56-64, Luxembourg.

Harrison, P., S. Abney, E. Black, D. Flickinger, C. Gdaniec, R. Grishman, D. Hindle, B. Ingria, M. Marcus, B. Santorini & T. Stralkowski. 1991. "Evaluating Syntax Performance of Parser/Grammars of English". *Proceedings of the Workshop on Evaluating Natual Language Processing Systems*, 71-77. Berkeley, Calif., U.S.A.

King, Paul J. 1989. *A Logical Formalism for Head-Driven Phrase Structure Grammar*. Ph.D. dissertation, Manchester University. Manchester, U.K.

King, Paul J. & Kiril Simov. 1998. "The Automatic Deduction of Classificatory Systems from Linguistic Theories". *Grammars* 1:2.103-153.

Kinyon, Alexandra & Owen Rambow. 2003. "The MetaGrammar: A Cross-Framework and Cross-Language Test-Suite Generation Tool". *4th International Workshop on Linguistically Interpreted Corpora*, 125-131. Budapest, Hungary.

Kübler, Sandra & Erhard Hinrichs. 2001. "From Chunks to Function-Argument Structure: A Similarity-based Approach". *29th Annual Meeting of the Association for Computational Linguistics* (*ACL-EACL'01*), 346-353. Toulouse, France.

Lin Dekang. 2003. "Dependency-based Evaluation of Minipar". *Treebanks. Building and Using Parsed Corpora* ed. by Anne Abeillé, 317-329. Dordrecht: Kluwer.

Oepen, Stephan, Ezra Callahan, Dan Flickinger & Chris Manning. 2002. "LinGO Redwoods: A Rich and Dynamic Treebank for HPSG". *Workshop Beyond PARSEVAL, Int. Conf. on Language Resources and Evaluation*, 17-22. Las Palmas, Spain.

Pollard, Carl J. & Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Chicago, Illinois: University of Chicago Press.

Simov, Kiril. 2002. "Grammar Extraction and Refinement from an HPSG Corpus". *ESSLLI Workshop on Machine Learning*, 38-55. Trento, Italy.

Simov, Kiril, Gergana Popova & Petya Osenova. 2002. "HPSG-based Syntactic Treebank of Bulgarian (BulTreeBank)". *A Rainbow of Corpora: Corpus Linguistics and the Languages of the World* ed. by Andrew Wilson, Paul Rayson & Tony McEnery, 135-142. Munich: Lincom-Europa.

Simov, Kiril, Milen Kouylekov & Alexander Simov. 2002. "Incremental Specialization of an HPSG-based Annotation Scheme". *Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data, Int. Conf. on Language Resources and Evaluation* (*LREC'02*), 16-23. Las Palmas, Spain.

Arabic Morpho-syntax for Text-to-Speech

ALLAN RAMSAY & HANADY MANSOUR
School of Informatics, Univ. of Manchester, Manchester, U.K.

Abstract

Attempts to provide text-to-speech systems for Arabic are considerably hampered by the fact that written Modern Standard Arabic omits a great deal of information about short vowels, and a certain amount of other phonetically relevant information. We argue that it is possible to recover this information by exploiting a mixture of morphological, syntactic and shallow semantic information. To do this effectively, the various levels of processing have to be carefully coordinated. We do that by delaying evaluation of constraints for as long as possible, so that we do not explore unnecessary hypotheses.

1 Outline

Written Modern Standard Arabic (MSA) omits all short vowels, and underspecifies a number of other phonetically relevant markers. This is somewhat inconvenient if you want to build a text-to-speech system for MSA, since the TTS cannot produce appropriate output unless this information is available (El-Shafei 2002, El-Imam 2001).

The work described below forms a preliminary step on the way to a TTS system for Arabic. Our goal is to reconstruct what we will call the FULL FORM of the short written form. This full form contains the diacritic markers which indicate the presence (or absence) of short vowels and other unwritten material. Mansour (2000) argues that full forms of this kind provide all the information you need in order to produce a phonetic transcription which can be used to drive a TTS. Certainly without it there is no hope of constructing such a system.

The problem with MSA is that almost every written form corresponds to a set of different words with different pronunciations – مدرسة (*mdrsT*) could be any of مدرسة (`madrasaT`)[1], مدرّسة (`mudarrisT`), مدرّسة(`mudarrasT`), كتب (*ktb*) could be any of كُتُبْ (`kutub`) (noun) كَتَبَ (`kataba`) (intransitive or transitive verb), كُتِبَ (`kutiba`) (passive of transitive كَتَبَ (`kataba`)), كَتَّبَ (`kattaba`) (transitive or ditransitive verb) or كُتِّبَ (`kuttiba`) (passive of either reading

[1] Where we write a standard written form of MSA, the transliteration is in *standard italics*; where we include the diacritics to show the underlying form, the gloss is in `fixed width font`, to emphasise that it is underlying forms of this kind that we are attempting to compute.

of كَتَّبَ (kattaba)), and so on. If we want to make a TTS system for MSA, we clearly need to make the right choices about which of these full forms was intended in a given sentence.

Our problems, then, are considerably more difficult than those faced by people who want to give an account of the morphology of full form Arabic (Kiraz 2001, McCarthy & Prince 1990), since we need an account of the structure of full form Arabic but we need to apply it in a situation where all we have is the written form, where short vowels and some consonants are omitted.

The approach we take to the task of obtaining full form Arabic words from written MSA, then, is as follows:

1. Reconstruct the full form of each written word in isolation, delaying any choices which the surrounding syntactic or semantic context can settle later (Section 2).
2. Parse the text (Section 3). As you construct syntactic analyses, check that any selection restrictions imposed on arguments or on the targets of modifiers are satisfied.
   As this step proceeds, choices which were left open when the words were looked at in isolation may be settled. We use the SICStus Prolog coroutining facilities (SICS 2002) to delay making decisions until the information needed to settle them deterministically is available.

2 Morphology

The critical part of the task is the determination of the possible full forms of a written word. Our approach has two major elements:

- We have to determine the morphemes that make up the words we are faced with – what is the root, what is the vocalic pattern, what are the affixes?
- We have to determine the full form that corresponds to the specific set of morphemes that make up the word in question.

The interactions between these two parts of the process are rather intricate. For simplicity we will describe the analysis of word structure first (Section 2.1) and the process of filling in the gaps second (Section 2.2).

2.1 *Categorial morphology*

Arabic words, like the words of most languages, are made up of sets of meaningful elements, where there is typically a root which carries the general semantic notion underlying the word, a derivational component that picks out a particular variant of this semantic essence, and a number of minor elements which carry information about matters such as tense, number and thematic role (generally

in the form of case markers). In many languages these elements are simply concatenated, but Arabic, like other Semitic languages, employs a nonlinear mixture of concatenation and superimposition of elements.

In the short, written form of MSA, the process of superimposing a vowel pattern on a consonantal root has very little visible effect. Where the vowel is in fact long then it is written, but much of the time the vowel pattern consists of short vowels or short pauses which are not written. A great deal of the information that we need in order to work out what these unwritten elements must have been is carried by the visible affixes which make it possible to determine whether, for instance, a particular word is a nominal or a verbal form, and to assign values for tense, number and agreement.

Kiraz (2001) notes that the description of the sequence of affixes that are required by a given root can be described by a simple unification grammar. It is well known that any such grammar is equivalent to a categorial grammar using the standard rules of categorial grammar given in Figure 1. We follow Bauer (1983) in using the formalism of categorial grammar, but we extend standard categorial grammar with the second pair of rules in Figure 1: these rules allow us extra flexibility when specifying the number of affixes that a given item requires, whilst allowing a strictly left←right approach to morphology.

```
% R ⟹ R/A, A
R(affixes=AFFIXES)
⟹ R(affixes=[A(dir=after) | AFFIXES]), A

% R ⟹ A, R\A
R(affixes=AFFIXES)
⟹ A, R(affixes=[A(dir=before) | AFFIXES])

% R/AFFIXES ⟹ R/A, A/AFFIXES
R(affixes=AFFIXES)
⟹ R(affixes=[A(dir=after)]), A[affixes=AFFIXES]

% R/AFFIXES ⟹ A/AFFIXES, R\A
R(affixes=AFFIXES)
⟹ A[affixes=AFFIXES], R(affixes=[A(dir=before)])
```

Figure 1: *Categorial rules*

The first move, then, is to use descriptions of this kind to capture the constituent structure of a typical Arabic word.

The general pattern is that a single root gives rise to a large number of different nouns and verbs, each of which requires a range of inflectional affixes such as tense, number, gender and case markers, possibly in conjunction with a set of

inflectional affixes.

In general, then, the dictionary entry for a root does *not* specify that it is a noun or a verb. Roots have no syntactic category assigned to them. Instead they *always* require a derivational affix[2] in order to fix the category. The basic form of an Arabic root, then, is

```
{syntax=S,
 affixes=[{syntax=S, affix=deriv}]}
```

This says that the first thing that any Arabic root does is to combine with a derivational affix, with which it is going to share all its syntactic properties. So if the derivational affix says it's a nominal affix then the whole thing is a noun, if it says it's a verbal affix then the whole thing is a verb.

So we can now just list all the roots and affixes in the lexicon, look them up as we work our way through the word, and combine them.

The problem here is that there are quite a large number of derivational affixes (we currently have about 75 nominal affixes and a smaller number of verbal ones), many of which are either empty or appear in the written form just as the letter مـ (*m*), but which all give rise to a different set of diacritics. Given that we are working with MSA, where the diacritics are not written but have to be determined by the program in order to drive the TTS, we can't tell which version of the derivational affix has been used in the construction of the surface form. We can't tell, for instance, whether the surface form مدرس (*mdrs*) was obtained using the version of مـ (*m*) which has full form مَ (ma) and which has the diacritic pattern ـْ (0)+ـُ (u) or the one that has full form مُ (mu) and diacritic pattern ـَ (a)+ ـَ (a) or ...

We do not want to have, say, 16 different prefixes all of which are written as مـ (*m*), where we have to try each in turn with a given root and then try all the ones that the root accepts in all the syntactic contexts where the word is used.

We therefore have a single lexical entry for each surface form of a derivational affix, and we allow the root to specify which variants of the affix, and hence which underlying form, it can accept. *The choice between different variations of the derivational affix is delayed until we know the syntactic and semantic context in which the word is being used.* There are, for instance, sixteen nominal prefixes that are written as مـ (*m*). Two of these combine with the root درس (*drs*), producing مُدَرِّس (mudarris) and مُدَرَّس (mudarras).

Once the derivational affix has fixed the basic category, we need to find any others that are required. Since the root doesn't know whether it is going to be a nominal or a verbal form, we can hardly expect it to specify what other affixes are required. Instead we exploit the freedom provided by Figure 1 to let the derivational affix specify what's required next.

[2] which may be empty, e.g., for pattern I verbs and for a wide range of nouns

We assume that Arabic nouns have gender, number and case markers, and that Arabic verbs have tense, gender, number and person markers, any of which may be empty or missing. Nominal derivational affixes therefore specify that a gender marker is required, so that the description of مـ (*m*) looks like

```
{form=مـ (m),
 affix=deriv
 {affixes=[{affix=gender, dir=after}]}]}
```

Similarly, we assume that verbal derivational affixes require a tense affix to be found, which will either be a prefix or empty. The tense affix may in turn demand an agreement affix which will fix both number and gender, so that يتكاتبون (*ytkātbwn*) consists of the root كتب (*ktb*), a verbal derivational suffix ت (*t*), a tense prefix يـ (*y*), and an agreement suffix ون (*wn*). The derivational suffix specifies the subcat for the verb (in this case that it is transitive) and demands a tense marker. The one that turns up, namely يـ (*y*), marks the verb as third person present tense and demands an agreement marker, and the agreement marker ون (*wn*) completes the description by marking it as masculine plural.

2.2 *Diacritics*

So we now know that يتكاتبون (*ytkātbwn*) is made out of four parts, which specify its syntactic properties. What we still need is to discover its full form يَتَكَاتَبُونَ (yatakātabuwna).

We start by considering a simpler case, and concentrating on the diacritics in the root. Consider يكتبون (*yktbwn*), where the derivational affix is empty.

We know that the root has holes in it, waiting to be filled, and that the fillers will differ depending on the derivational affix and, if it turns out to be a verb, the tense. We know that كتب (*ktb*) belongs to a fairly widespread class of verbs for which the past diacritics are ـَ (a)+ـَ (a) and the present diacritics are ـُ (0)+ـُ (u). We therefore encode this information in the root, and let the tense affix select which of the possible sets of diacritics is to be used on this occasion.

```
{wform=كتب (ktb), uform=ك؟ت؟ب (k?t?b),
 presprefix=ـَ (a),
 diacritics=(present=ـُ (0)+ـُ (u), past=ـَ (a)+ـَ (a))}
```

We have split the `form` into two parts – the short written form `wform` and the full underlying form `uform`. The full form has place-holders, which can be filled in with short or long vowels, or left empty, as appropriate. The `diacritics` specifies how the holes are to be filled in under specified circumstances.

The present prefix then selects the appropriate set of diacritics, and notes that what we have is the present tense form of the verb:

```
{wform=ـيـ (y), uform=X+ـيـ (y+X),
 cat=verb,
 presprefix=X,
 diacritics=(present=PRESENT,past=???,actual=PRESENT)}
```

Unifying these leads to

```
{wform=يكتب (yktb),
 uform=ـيـ+ـَـ+كـ?ـتـ?ـبْ (y+a+k?t?b),
 presprefix=ـَـ (a),
 diacritics=(present=ـَـ (0)+ـُـ (u), past=ـَـ (a)+ـَـ (a),
             actual=ـَـ (0)+ـُـ (u))}
```

from which it is easy to obtain the proper full form يَكْتُبُ (yaktubu) by substituting the selected diacritics in order and deleting 0's and +'s.

3 Syntax

It seems, then, that it is possible to determine a range of possible underlying forms of a written MSA word, but that there is likely to be considerable ambiguity as to which form was intended in a given context. The next step is to parse the input text to eliminate readings of individual words that do not contribute to globally acceptable readings of the whole text.

We use an HPSG-like grammar with a parsing algorithm which has been tuned for dealing with languages where phrase order is fairly unconstrained (Ramsay 1999, Ramsay 2000). Consider the short sentence (1)

(1) درس دارس الدرس. (*drs dārs aldrs.*)

درس (*drs*) and دارس (*dārs*) each have multiple interpretations – درس (*drs*) has one nominal and two verbal interpretations, as does دارس (*dārs*). There are, therefore, nine possible combinations of underlying forms. The sentence as a whole,y however, has just two legal structural analyses – the one in Figure 2 and an exactly parallel one with دَرَّسَ (darrasa) as the verb.

The only readings we obtain have one of the verbal forms of درس (*drs*) as the verb, دارس (*dārs*) as subject and الدرس (*aldrs*) as object (i.e., with VSO word-order). What happened to all the other potential analyses?

- We don't get an SVO reading with دارس (*dārs*) as verb and درس (*drs*) as subject because indefinite subjects cannot occur in SVO order sentences.

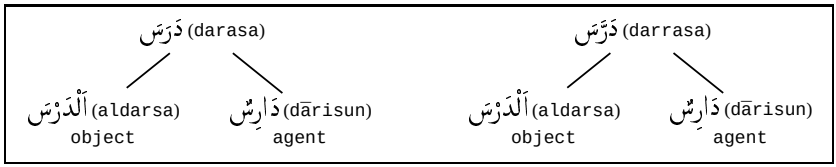

Figure 2: درس دارس الدرس drs dārs aldrs.

- We don't get a VOS reading with دارس (*dārs*) as object and الدرس (*aldrs*) as subject or an OVS reading with درس (*drs*) as object and دارس (*dārs*) as verb because for that to happen either دارس (*dārs*) and درس (*drs*), as indefinite singular masculine nouns, would need an explicit accusative case marker, which neither of them has.
- We don't get a reading with the intransitive reading of درس (*drs*) because there is then no role for الدرس (*aldrs*).

We see, then, that we can weed out candidate forms for MSA written words by checking to see whether they contribute to syntactically well-formed sentences. We may still have a number of analyses left, as above. If they all have the same phonetic transcription then we don't need to choose between them, since our task is to obtain the information required for TTS, but if we do have to make a choice then we rely on simple selection restrictions of the kind pioneered by Wilks (1978).

Obtaining a syntactic analysis of the text, then, is important because it helps rules out many possible forms of individual words (and is also crucial if we want to use information about the meaning of the utterance to eliminate candidate interpretation, since the syntactic structure underlies the semantic analysis). Given our task of producing the information needed for text-to-speech, however, it has an additional role: in at least some, more formal, registers, Arabic nouns have case markers, which should be pronounced but are not written.

This is problematic. Case markers are unwritten, but their underlying form is governed by a fairly complex set of constraints. The following provides a rough summary of some of the more important rules:

- The form of the case marker depends on the case, the number and gender of the noun, and on whether the noun is definite or indefinite.
- One of the arguments of a verb gets assigned the role of subject. The subject will be nominative if the verb is tensed (i.e., if it is a main sentence or the clause containing it is governed by أن (*ʔan*)), and accusative if the verb is untensed (i.e., if it is governed by أنّ (*ʔann*) or إنّ (*ʔinn*)) (Mohammed, 2000).
  Nominal clauses, of course, do not have verbs, so we can hardly say that

the case of the subject is governed by the tense of the verb. The case of the subject here is determined by the status of the clause as a whole – if it's a main clause or governed by أن (*ʔan*), then the subject will be nominative, if it's governed by أنّ (*ʔann*) or إنّ (*ʔinn*) then the subject will be accusative.

- Other arguments will be accusative (objects and auxiliary objects) or marked by a preposition.
- The satellite element in a construct NP will be genetive. The case marker on the nucleus will have the definite form, even though the nucleus does *not* have a definite article attached to it: this is because the construct NP as a whole is definite, and the nucleus inherits this property.

(2) illustrates a number of these points.

(2) أعتقد الكاتب أنّ الكتاب في المكتب. (*ʔaʿtaqad alkātb ʔann alktāb fy almktb.*)

The subject of (2) is الكاتب (*alkātb*). Because it is the subject of a main clause it is assigned nominative case, and because it is masculine definite singular the underlying form becomes اَلْكَاتِبُ (alkātibu). The subject of the embedded nominal sentence is الكتاب (*alktāb*). Because the embedded sentence is governed by أنّ (*ʔann*), الكتاب (*alktāb*) is assigned accusative case, which means that its underlying form is اَلْكِتَابَ (alkitāba).

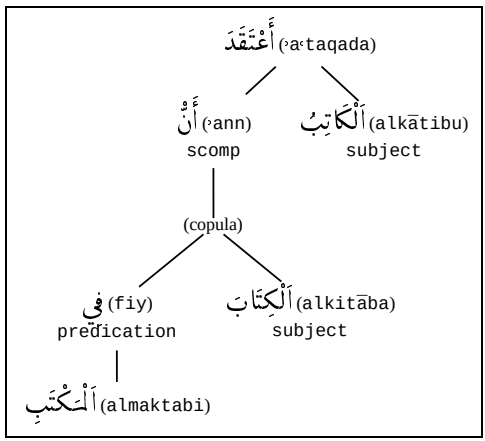

Figure 3: أعتقد الكاتب أنّ الكتاب في المكتب. ʔtqd alkātb ʔann alktāb fy almktb.

(3) shows what happens with construct NPs.

(3) كتاب العالم في المكتب. (*ktāb al-ʿālm fy almktb.*)

The satellite العالم (*al-ʿālm*) is marked as genetive, and hence has the underlying form اَلْعَالِمِ (al-ʿālimi). The head كتاب (*ktāb*) is the subject of the whole sentence, and hence is nominative. However, although it appears to be indefinite, since it has no definite article attached to it, the fact that it is the head of a construct NP makes it definite, and hence its underlying form is كِتَابُ (kitābu) rather than كِتَابٌ (kitābun) as you would expect for a masculine indefinite singular nominative.

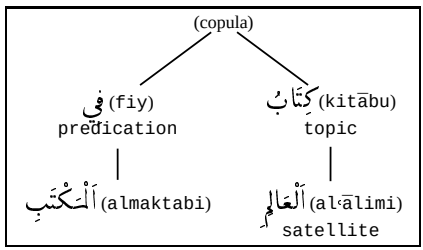

Figure 4: كتاب العالم في المكتب ktāb al-ʿālm fy almktb

4 Conclusions

The system described above computes the underlying full forms for input MSA texts where short vowels and other phonetically significant material is missing from the written form. In order to do this, we carry out the following processes:

- morphotactic/morphophonemic analysis, using a categorial description of how roots and affixes (*including* circumfixes) combine.
- slot-and-filler insertion of diacritics, where the choice of diacritic pattern is determined by the morphotactic analysis and the surrounding syntactic and semantic context.
- syntactic analysis using a grammar which deals with the range of word orders that are permitted in simple Arabic sentences, and which also covers the intricate case marking constraints associated with nominal sentences and construct phrases.
- semantic analysis: this is not discussed at length in the current paper for reasons of space, but it is used to provide further help with disambiguation.

We are currently integrating the system with the MBROLA speech synthesiser (Dutoit 1996), using the Arabic diphone set supplied by Nawfal Tounsi. The

output is currently intelligible, but extremely flat. The structural analysis obtained by the system, however, enables us to split the text globally into 'phonetic phrases' and locally into syllables. This work is still fairly preliminary – we can use the system to drive the synthesiser, we can suggest pitch contours (which the synthesiser can respond to), but it has to be said that we are still some way from producing natural sounding Arabic.

REFERENCES

Bauer, Laurie. 1983. *English Word Formation*. Cambridge: Cambridge University Press.

Dutoit, Thierry, Vincent Pagel, Nicolas Pierret, F. Bataille & O. van der Vreken. 1996. "The MBROLA Project: Towards a Set of High-quality Speech Synthesizers Free of Use for Non-commercial Purposes. *Proceedings of the International Conference on Spoken Language Processing* (*ICSLP'96*), vol. 3, 1393-1396. Philadelphia, Pennsylvania.

El-Imam, M. A. 2001. "Synthesis of Arabic from Short Sound Clusters". *Computers, Speech and Language* 15:4.355-380.

El-Shafei, Moustafa, Husni Al-Muhtaseb & Mansour Al-Ghamdi. 2002. "Techniques for High Quality Arabic Speech Synthesis". *Information Sciences* 140:3/4.255-267.

Kiraz, George Anton. 2001. *Computational Nonlinear Morphology: With Emphasis on Semitic Languages*. Cambridge: Cambridge University Press.

Mansour, Hanady. 2000. *Phonetic Transcription Rules for Arabic Text-to-Speech*. M.A. thesis, University of Alexandria, Alexandria, Egypt.

McCarthy, John & Alan Prince. 1990. "Prosodic Morphology and Templatic Morphology". *Perspectives on Arabic Linguistics II: Papers from the 2nd Annual Symposium on Arabic Linguistics* ed. by Mushira Eid & John McCarthy (= *Amsterdam Studies in the Theory and History of Linguistic Science IV: Current Issues in Linguistic Theory*, 72), 1-54, Amsterdam. Benjamins.

Mohammed, Mohammed A. 2000. *Word Order, Agreement and Pronominalisation in Standard and Palestinian Arabic*. (= *Current Issues in Linguistic Theory*, 181). Amsterdam & Phyladelphia: John Benjamins.

Ramsay, Allan M. 1999. "Direct Parsing with Discontinuous Phrases". *Natural Language Engineering* 5:3.271-300.

Ramsay, Allan M. & Helen Seville. 2000. "Unscrambling English Word Order". *Proceedings of the 18th International Conference on Computational Linguistics* (*COLING-2000*) ed. by Martin Kay, 656-662. Saarbrücken: Universität des Saarlandes.

SICS. 2002. *SICStus Prolog manual*. Stockholm: Swedish Institute for Computer Science.

Wilks, Yorick. 1978. "Making Preferences More Active". *Artificial Intelligence* 11:3.197-223.

Guessing Morphological Classes of Unknown German Nouns

PRESLAV NAKOV*, YURY BONEV**, GALIA ANGELOVA***,
EVELYN GIUS**** & WALTHER V.HAHN****

*University of California at Berkeley
**Sofia University
***Bulgarian Academy of Sciences
****University of Hamburg

Abstract

The MorphoClass system for recognition and morphological classification of unknown German words is described. It learns and applies ending guessing rules, similar to those proposed for POS guessing. Then, given raw texts, it outputs the unknown nouns together with hypotheses about their possible stems and morphological class(es). The system's design and implementation are presented and demonstrated by means of extensive evaluation.

1 Introduction

The efficient processing of unknown words is a primary problem for every *Natural Language Processing* (NLP) system. No matter how big its lexicon is, it will always face unknown wordforms[1]. Existing systems either use spell-checkers, lists of exceptions and gazetteers of proper names, or rely on data-driven approaches in order to model these phenomena and decide on the type and category of unknown wordforms.

While most of the unknown words processing systems focus on the prediction of the most likely *part of speech* (*POS*), **MorphoClass**[2] targets the *morphological class* of unknown nouns. It groups unknown German input wordforms as candidates for a single paradigm and tries to propose a *stem* and a *morphological class*. We define the *stem* as the common part shared by all inflected wordforms (up to valid alternations). Together with the morphological class, it determines all wordforms that could be obtained by an inflection of the same base paradigm. **MorphoClass** solves the "guessing" problem as a sequence of subtasks including: (*i*) identification of unknown words (at present, limited to nouns); (*ii*) recognition and grouping of the inflected forms of a word; (*iii*) compound splitting; (*iv*) morphological stem analysis; (*v*) stem hypothesis generation

[1] Misspelled words and orthographic variants from e.g., the recent reform of German orthography also result in "new words".

[2] **MorphoClass** was developed within the EC funded project "BIS-21 Centre of Excellence" ICA1-2000-70016 and was additionally supported by the bilateral cooperation scheme of Hamburg University and Sofia University.

for each group of inflected forms; (*vi*) ranking the list of hypotheses for possible morphological classes for each word group.

This is a multiple-stage process, which relies on: (*i*) **morphology** – compound splitting, inflection, affixes; (*ii*) **global context** – wordforms from the whole input, word frequency statistics, ending guessing rules etc.; (*iii*) **local context** – adjacent words: articles, prepositions, pronouns; (*iv*) **external sources** – lexicons, German grammar information etc.

MorphoClass is not a POS guesser as it is not restricted to the local context. The system is a kind of morphological class guesser, which could (but does not need to) work after or before a POS tagger has marked the unknown nouns. **MorphoClass** can also be used as a lemmatiser, as it outputs the stem and the morphological class for each known word. The system is not a stemmer since it does not conflate derivational forms: e.g., *generate* and *generator* would be grouped together by most stemmers but not by **MorphoClass**. To the best of our knowledge, this is the only system that attempts to address these morphological issues.

The paper is organised as follows. Section 2 comments on related work, Section 3 discusses the system's resources and Section 4 shows the system at work. Section 5 presents the evaluation, Section 6 points to some further improvements by linear context consideration and Section 7 contains the conclusion.

2 Related work

Our approach is related to and influenced by some classical NLP tasks, such as morphological analysis and POS tagging. See (Nakov et al. 2003) for a detailed comparison to these and other tasks. *German morphology:* (Adda-Decker & Adda 2000) propose rules for morpheme boundary identification, after the occurrence of sequences like: $-ungs$, $-hafts$, $-lings$, $-tions$, $-heits$. The problem of German compound splitting is considered in depth by (Goldsmith 1998; Lezius 2000; Ulmann 1995). *General morphology:* (Goldsmith 2001) performs a minimum description length corpora-based analysis of the morphology of several European languages. A very influential is the work of Brill (Brill 1997) who builds linguistically motivated rules using both a tagged corpus and a lexicon. He looks not only at the affixes but also checks their POS in a lexicon. Mikheev proposes a similar approach but learns the ending guessing rules from a raw text (Mikheev 1997). (Daciuk 1999) speeds up the process using finite state transducers. We are not aware of any other system that guesses morphological classes by observing the endings only, without considering any word formation rules.

3 Resources: lexicons and grammatical knowledge

A word is considered *known* by **MorphoClass**, if it is contained in one of its lexicons. The *Stem Lexicon* (*SL*) is compiled from both the NEGRA corpus (NEGRA 2001) and the full-form Morphy lexicon (Lezius 2000), and currently contains 13,147 German nouns. SL facilitates compound recognition since the compound splitting module relies on SL's noun stems. The *Expanded Stem Lexicon* (*ESL*) includes all wordforms derived from the SL entries and is used during the learning stage for the ending guessing rules. The *Word Lexicon* (*WL*) contains adjacent closed-class words such as articles, pronouns, etc.

Class	**Singular**				**Plural**			
	nom	**gen**	**dat**	**akk**	**nom**	**gen**	**dat**	**akk**
m1	0	[e]s(1)	[e]	0	e	e	en	e
...	...	...	...	...	...	...	...	...
m3a	0	[e]s(1)	[e]	0	er	er	ern	er
...	...	...	...	...	...	...	...	...
m9	0	[e]s(1)	[e]	0	en	en	en	en
...	...	...	...	...	...	...	...	...
n20	0	[e]s(1)	[e]	0	e	e	en	e
n21	0	[e]s(1)	[e]	0	er	er	ern	er
...	...	...	...	...	...	...	...	...
n25	0	[e]s(1)	[e]	0	en	en	en	en

Table 1: *Morphological classes of German nouns*

The **MorphoClass** morphological classes have been designed for the DB-MAT system (DB/R/-MAT 1992-1998). Each class contains 8 noun endings (see Table 1) appended to the stem with possible alternations and other changes expressed by rules. E.g., the ending *-s* for *Genitive Singular* is appended to stems from class **m1**, after adding a preceding *-e-* and taking into account *rule 1*, which encodes: "when the basic nominative form ends by *s/sch/x/chs/z/tz/*... the vowel *-e-* is obligatory". For a complete list of the morphological classes and rules see (Nakov et al. 2002).

We will focus now on the processing of unknown nouns only (the known ones are identified by lexicon lookup). The successful recognition of unknown nouns substantially depends on the fact that the German nouns are capitalised (with very few exceptions, each capitalised word is either a noun, a named entity or starts a sentence). **MorphoClass** produces one of the following judgements for each group of wordforms sharing a stem:

- COMPOUND – successfully split using the available lexicon, so the morphological class of the last word in the compound is assigned;
- ENDING RULE – ending guessing rule applied, class assigned;

- NO INFO – no decision, incompatible classes elimination only.

For the morphological class prediction we adopted an ending guessing rules mechanism, which has been originally proposed for POS guessing (Mikheev 1997). We built 482 rules when running the rule induction against the SL and 1,789 rules when the SL entries were weighted according to their frequencies in a raw text (see Table 2). We considered all endings up to 7 characters long and met at least 10 times in the training raw text, provided that there were at least 3 characters remaining to the left, including at least one vowel. For each noun we extracted all possible suffixes (e.g., from *Vater* we obtain *-r*, *-er* and *-ter*) and for each ending – a list of the morphological classes it appeared with and their corresponding frequencies. It is intuitively clear that a good ending guessing rule would be: unambiguous (predicts a particular class without or with only few exceptions); frequent (must be based on a large number of occurrences); long (the longer the ending, the lower the probability that it will appear by chance, and thus the better its prediction). While the maximum likelihood estimation is a good predictor of the rule quality, it does not take into account the rule length or the rule frequency. So, following (Mikheev 1997) we adopted the score:

$$\text{score} = p - \frac{t^{n-1}_{(1-\alpha)/2}\sqrt{\frac{p(1-p)}{n}}}{1+\log l},\ p = (x+0.5)/(n+1)$$

where: l is the ending length, x is the number of successful rule guesses, n is the total number of training instances compatible with the rule, p is a smoothed version of the maximum likelihood estimation, which ensures that neither p nor $(1-p)$ could be zero, $\sqrt{\frac{p(1-p)}{n}}$ is an estimation of the dispersion, $t^{n-1}_{(1-\alpha)/2}$ is a coefficient of the t-distribution (with $n-1$ degrees of freedom and confidence level α).

Ending	**Confidence**	**Predicted class(es)**	**Class frequency(s)**
heit	0.999496	f17	1761
nung	0.999458	f17	1638
schaft	0.999427	f17	1439
keit	0.999412	f17	1510
chaft	0.999409	f17	1439
tung	0.999408	f17	1498
gung	0.999394	f17	1464
haft	0.999383	f17	1439
lung	0.999182	f17	1084

Table 2: *The most confident ending guessing rules (learned from the lexicon and weighted on a raw text).*

We keep the rules whose score is above some threshold. Currently, we use 0.90, which produces high quality rules. This makes the system conservative as no

specific morphological class (out of the feasible ones for the target group of wordforms) is proposed unless **MorphoClass** is confident enough in its choice. If more texts are presented to the system, it would possibly observe more wordforms of the target unknown noun and will rule out some of the possible classes. If we want a choice at any price, we can move the threshold down or allow ambiguous ending guessing rules that predict more than one morphological class (as Mikheev did).

4 Examples

MorphoClass attempts to generate a stem for each group, as shown in Table 3. For each stem in column one it checks the existence of a morphological class that may generate all wordforms of column three. If at least one is found, **MorphoClass** accepts the current coverage as feasible, otherwise it tries to refine it in order to make it acceptable. It is possible that the same stem is generated by a set of words that otherwise could not be covered together. In the first step it does not matter whether the stem is correct, but just whether there is a morphological class that could generate all the forms in column 3.

Stem	**#**	**Wordforms covered**
Haus	7	{ Haus, Hause, Hausen, Hauses, Hausse, Hauser, Hausern }
Groß	6	{ Große, Großen, Großer, Großes, Große, Großen }
Große	6	{ Große, Großen, Großer, Großes, Große, Großen }
Spiel	**6**	{ **Spiel, Spiele, Spielen, Spieler, Spielern, Spiels** }
Ton	6	{ Ton, Tonnen, Tons, Tonus, Tone, Tonen }
Geschäft	5	{ Geschäft, Geschäfte, Geschäften, Geschäftes, Geschäfts }
Schrei	3	{ Schrei, Schreien, Schreier }

Table 3: *Largest wordform sets before the stem refinement*

For example, in Table 3, we do not reject the stem *Spiel*, which is incompatible with the set of wordforms. We decide that *Spiel*, *Spiele*, *Spielen* and *Spiels* are correct members of the *Spiel* paradigm, while *Spieler* and *Spielern* are not and probably belong to a different one. *Spiel*, *Spiele*, *Spielen* and *Spiels* might be generated from *Spiel* by the four classes *m1*, *m9*, *n20* and *n25*, while *Spieler*, *Spielern* – may be generated from *Spiel* by *m3a* and *n21* (see Table 1). Thus, both groups are acceptable, taken separately. The first group is bigger and therefore more likely, i.e., the first four wordforms belong to the paradigm of *Spiel*. Applying ending guessing rules, we will have to choose between four possible morphological classes: *m1*, *m9*, *n20* and *n25*. For *Spieler* and *Spielern* **MorphoClass** will continue to explore other possible stems. In case of same cardinality of the two groups, we would prefer the most likely morphological class, according to Mor-

phy's lexicon and NEGRA. In the worst case **MorphoClass** would propose two candidates.

Table 4 shows the interaction between compound splitting and ending guessing. It is obtained from Table 3 after several iterations of stem refinement. The compound *Bildungsurlaub* (educational holiday) in the first row is composed of *Bildung* (study) and *Urlaub* (vacation). The last noun *Urlaub* determines both the compound gender (masculine) and its morphological class. However, a plural form of this paradigm *Bildungsurlaube* (last row) covers the base form *Laube* (summer house), which is a feminine noun. The first and the last rows include *Bildungsurlauber* (person in study leave), which does not belong to these paradigms.

In the first step, **MorphoClass** identifies the valid compounds together with the known nouns and morphotactic rules. Here it will see the stem *Bildungsurlaub* first, as it is suggested by the 4 wordforms, and will find that *Bildung* as well as *Urlaub* are in the lexicon, i.e., *Bildung-s-urlaub* is a valid compound. However, it does not generate *Bildungsurlauber* as a member of the same paradigm. At this moment *Bildung-s-urlaub* will be kept as a compound and *Bildungsurlauber* as a single form that may belong to another paradigm.

Unknown stem	**#**	**Words that generated the stem**
Bildungsurlab	**4**	{ **Bildungsurlaub, Bildungsurlaube, Bildungsurlauben, Bildungsurlauber** }
Ortsbezirk	4	{ Ortsbezirk, Ortsbezirke, Ortsbezirken, Ortsbezirks }
Bildungsurlaube	**3**	{ **Bildungsurlaube, Bildungsurlauben, Bildungsurlauber** }

Table 4: *Unknown stems ordered by the number of word tokens covered*

With *Bildungsurlaube* in the last row of Table 4, **MorphoClass** will try to decompose it as *Bildungsur-laube*, since *Laube* is in the lexicon as well[3]. Since *Bildungsur-laube* fails, **MorphoClass** will connect the first two forms in the last row to the paradigm in the first row since *Bildungsurlaub* was treated as a valid component.

In case neither *Bildung*, *Urlaub* nor *Laube* are in the lexicon, **MorphoClass** will apply ending guessing rules to *Bildungsurlaub*, find a morphological class and exclude *Bildungsurlauber* from the paradigm as impossible. Finally, **MorphoClass** will not treat *Bildungsurlaube* as a possible stem since it is already covered by *Bildungsurlaub*. Thus **MorphoClass** will not guess that *Bildung*, *Urlaub*, and *Laube* are possible base forms of German nouns.

[3] But it will not find *Bildungsur-*. Moreover, *Laube* has no form *Lauber* and thus *Bildungsurlauber* cannot be generated.

5 Evaluation

The **MorphoClass** system has been manually evaluated with the following texts: (*i*) **Kafka:** *Erzählungen*, 3,510 word types, 13,793 word tokens; (*ii*) **Goethe 1:** *Die Wahlverwandtschaften*, 10,833 types, 79,485 tokens; (*iii*) **Goethe 2:** *Wilhelm Meisters Lehrjahre*, 17,252 types, 194,266 tokens.

Table 5 summarises **MorphoClass'** performance. The compound splitting rules have a coverage of 32% and their precision is over 92% for all text types. A substantial amount of the remaining stems are covered by the ending guessing rules, which are applied in over 40% of the cases, but their precision in isolation was lower (details follow). However, **MorphoClass** has no dictionary of named entities and its ending guessing rules were trained on the small lexicon of Morphy, where nominalised verbs are listed as nouns. So we do not pretend that the ending guessing rules are based on representative statistics. **MorphoClass** always produces a list of candidate classes but with "no info" it does not choose any of them. All results should be considered as relative to the available resources. Better performance can be achieved with a list of named entities and a broader initial lexicon. During the evaluation, the produced stems have been categorised as follows:

- **SET:** a *set* of classes is assigned instead of a single one;
- **PART:** a *correct* class is discovered but *not all* the correct ones;
- **WRONG:** a single class is assigned but it is *wrong*;
- **YES:** a single class is assigned and it is the only correct one;
- **SKIP:** the stem has been excluded from the current manual evaluation (proper names, non-German nouns, non-nouns or wrong stem).

We define *precision* and *coverage* as follows:

precision1 = YES / (YES + WRONG + PART)
precision2 = (YES + (scaled_PART)) / (YES + WRONG + PART)
precision3 = (YES + PART) /(YES + WRONG + PART)
coverage = (YES + WRONG + PART) / (YES + WRONG + PART + SET)

Coverage shows the proportion of the stems whose morphological class has been found, while *precision* reveals how correct it was. A scaling is performed according to the proportion of possible classes guessed g to the number of classes: if a stem belongs to k ($k > 2$) classes and **MorphoClass** found g of them (it finds exactly one in case of ending guessing rule but in case of compound splitting or no rule applicable, it might find more) *precision1* considers this as a failure (i.e., 0), *precision2* counts it as a partial success (will count it as g/k) and *precision3* accepts it as a full success (i.e., 1).

Table 6 shows the results of the manual evaluation using the above measures. We see that for the longer texts the precision is higher but the coverage is lower. We considered two baselines: *baseline 1* always predicts class *f16*, since it is the

	Nouns	**Compounds**	**Ending guessing**	**"No info" stems**
Kafka	473	185 (39.11%)	190 (40%)	98 (21%)
Goethe 1	1,706	551 (32.30%)	837 (49%)	318 (19%)
Goethe 2	2,838	896 (31.57%)	1,274 (45%)	668 (23%)

Table 5: *Noun wordforms in the different texts*

most frequent one, while *baseline 2* proposes the most frequent class but limited to the ones compatible with the proposed stem and the wordform group. Table 7 shows the performance of **MorphoClass** is well above both baselines.

	Kafka	*Goethe 1*	*Goethe 2*
Coverage	88.99%	84.80%	82.43%
Precision 1	74.23%	75.75%	82.36%
Precision 2	76.08%	77.21%	83.42%
Precision 3	81.44%	79.96%	85.22%

Table 6: *Evaluation results*

	Baseline 1			**Baseline 2**		
	Kafka	*Goethe 1*	*Goethe 2*	*Kafka*	*Goethe 1*	*Goethe 2*
Coverage	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Prec. 1	13.76%	9.61%	10.82%	16.40%	19.30%	21.29%
Prec. 2	13.76%	9.65%	10.86%	16.40%	19.34%	21.41%
Prec. 3	13.76%	9.69%	10.91%	16.40%	19.37%	21.56%

Table 7: *Baselines 1 and 2*

A more detailed evaluation has been performed on Kafka's *Erzählungen*. As Table 8 shows, the compound splitting rules have a very high precision: 93.62% (no partial matching: all the rules predicted a single class, even if more than one splitting was possible) and coverage of 43.12%. Ending guessing rules have a much lower precision: 56% for *precision 1* and 70% for *precision 3*. This is an overall coverage of 88.99% and precision of 74.23% (*precision 1*), 76.08% (*precision 2*) and 81.44% (*precision 3*). Note that the cascade algorithm is "unfair", since ending guessing rules have been applied only if the compound splitting rules had failed. So we made a second run with the compound splitting rules disabled and we obtained much higher coverage (76.15%) and precision (66.27%, 68.43%, 74.70%). Note that some of the stems are short and thus the ending guessing rules might act as compound splitting. This explains the improvement for the ending guessing rules on the second run.

	RUN 1			RUN 2
	compound splitting	**ending guessing rules**	**overall (cascade)**	**ending guessing rules *only***
Coverage	43.12%	45.87%	88.99%	76.15%
Precision 1	93.61%	56.00%	74.23%	66.27%
Precision 2	93.62%	57.47%	76.08%	68.43%
Precision 3	93.62%	70.00%	81.44%	74.70%

Table 8: *Detailed evaluation on Kafka. The coverage is higher than in Table 5, since the NO INFO column is split into SET, PART and SKIP*

6 Improvement by linear context

MorphoClass does not consider all successfully guessed morphological classes to be equally likely. Its last module uses the two-word left context of the target noun and statistical observations from the NEGRA corpus to produce a ranking of the feasible morphological classes. The context words are limited to the articles, prepositions and pronouns, which can be used as a left predictor of the gender, case and number of the target noun. For instance, according to NEGRA, *eine* is most often followed by a feminine noun in Akk/Sing.: (Fem/Akk/Sg: 0.6714), (Fem/Nom/Sg: 0.3213) and (Fem/Dat/Sg: 0.0073).

NEGRA allowed us to acquire good left context statistics for about 75% of the nouns it contains (about 6% had no left context of the kind we needed). Our investigation shows that:

- The morphological class predicted by the left context rules coincides with the gender of the three most likely classes proposed by **MorphoClass** in 60% of the cases;
- In 78% of the cases, the morphological class predicted by the left context rules is among the classes produced by **MorphoClass**;
- The linear context improves the performance in about 14% of the cases.

7 Conclusion and future work

MorphoClass is a useful tool for lexical acquisition: it identifies new wordforms in the raw text, derives some properties and performs a morphological classification. It is clear that 100% accuracy is impossible, but the more wordforms it collects the better the guess will be. An important feature of **MorphoClass** is that its performance can be improved using bootstrapping by extending the lexicons with the newly acquired unknown wordforms, so that better results can be achieved incrementally. As new wordforms are collected from the whole text (or from a collection of texts) in a context-independent way, **MorphoClass** can be used as a lexicon-acquisition tool for automatic dictionary extension.

We tried to apply the same procedure for guessing morphological classes of unknown nouns in Bulgarian. The results were much worse (success rate of less than 50%) possibly due to the richer inflectional morphology and to the difficulties to identify unknown nouns in raw texts. The relatively high precision for German substantially depended on the fact that this language requires capitalisation of the nouns.

Possible directions of future development of **MorphoClass** include: refining the ending guessing rules (using a bigger lexicon), testing its feasibility in a lexical acquisition environment for German and application to other languages with relatively compact and regular morphology and possibly to other parts of speech.

REFERENCES

Adda-Decker, Martine & Gilles Adda. 2000. "Morphological Decomposition for ASR in German". *Workshop on Phonetics and Phonology in Automatic Speech Recognition*, 129-143. Saarbruecken, Germany.

Brill, Eric. 1997. "Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging." *NL Processing Using Very Large Corpora.* Kluwer Academic Press.

Daciuk, Jan. 1999. "Treatment of Unknown Words". *Workshop on Implementing Automata*, IX-1–IX-9. Potsdam, Germany.

DB-MAT/DBR-MAT. `nats-www.informatik.uni-hamburg.de/~dbrmat/`

Goldsmith, John. 2001. "Unsupervised Learning of the Morphology of a Natural Language". *Computational Linguistics* 27:2.153-198.

Goldsmith, John & Tom Reutter. 1998. "Automatic Collection and Analysis of German Compounds". *Proceedings of the The Computational Treatment of Nominals Workshop* (*COLING-ACL'98*) ed. by F. Busa, I. Mani & P. Saint-Dizier, 61-69. Montréal, Canada.

Lezius, Wolfgang. 2000. "Morphy – German Morphology, Part-of-Speech Tagging and Applications". *9^{th} EURALEX International Congress* ed. by U. Heid, S. Evert, E. Lehmann & C. Rohrer, 619-623. Stuttgart, Germany.

Mikheev, Andrei. 1997. "Automatic Rule Induction for Unknown Word Guessing". *Computational Linguistics* 23:3.405-423.

Nakov, Preslav, Yury Bonev, Galia Angelova, Evelyn Gius & Walther von Hahn. 2003. "Guessing Morphological Classes of Unknown German Nouns". *Proceedings of Recent Advances in Natural Language Processing* (*RANLP-2003*), 319-326. Borovets, Bulgaria.

Nakov, Preslav, Galia Angelova & Walther von Hahn. 2002. "Automatic Recognition and Morphological Classification of Unknown German Nouns". Bericht 243, FBI-HH-B-243/02, Universität Hamburg, Germany.

NEGRA corpus. 2001. `www.coli.uni-sb.de/sfb378/negra-corpus`

Ulmann, Martin. 1995. "Decomposing German Compound Nouns". *Recent Advances in Natural Language Processing* (*RANLP-1995*), 265-270. Tzigov Chark, Bulgaria.

Building Sense Tagged Corpora with Volunteer Contributions over the Web

RADA MIHALCEA* & TIMOTHY CHKLOVSKI**

*University of North Texas
**USC Information Sciences Institute (ISI)

Abstract

It is generally agreed that the success of a Word Sense Disambiguation (WSD) system depends, in large, on having enough sense annotated data available at hand. We report a Web-based approach to constructing large sense tagged corpora by exploiting agreement ofWeb users who contribute word sense annotation. We investigate the quantity and quality of the sense tagged data collected with this approach, and show that it compares favorably with the traditional method of hiring lexicographers.

1 Introduction

Onenotoriously difficult problem in understanding text is Word Sense Disambiguation (WSD).While humans usually do not even notice the word ambiguities when interpreting text, word ambiguity is very common, especially among the most frequent words. Despite much recent work on machine WSD algorithms within the recent SENSEVAL evaluation frameworks and elsewhere, there has not been as much progress on a related problem known to have a strong impact on the quality of WSD systems, namely the availability of sense tagged data.

With a handful of tagged texts currently available, existing WSD systems are able to deal only with few pre-selected words for which hand annotated data was provided. Out of the total of 20,000 English words which carry more than one possible meaning, currently available sense tagged data is limited to annotated examples for at most 300-400 words.

The rest of the chapter is organized as follows. In Section 2, we describe the system used to collect annotated data from Web users. In Section 3 and 4 respectively, we investigate the quantity and quality of data collected over the Web, and evaluate the WSD performance that can be achieved by relying on these data. We summarize our contributions in Section 5.

2 Building sense tagged corpora with the help of Web users

To overcome the current lack of sense tagged data and the limitations imposed by the creation of such data using trained lexicographers, we designed a system that enables the collection of semantically annotated corpora over the Web.

Sense tagged examples are collected using a Web-based application, *Open Mind Word Expert* (OMWE), that allows contributors to annotate words with their meanings. OMWE and the data it has collected are available online from `http://teach-computers.org` at the time of writing. Tagging is organized by word: for each ambiguous word for which a sense tagged corpus is desired, contributors are presented with a set of natural language (English) sentence-long contexts each of which includes an instance of the ambiguous word.

The overall process proceeds as follows. Initially, example sentences are extracted from a large textual corpus. If other training data for a given word is not available, a number of these sentences are presented to the users for tagging in *Stage 1*. Next, this tagged collection is used as training data for a WSD algorithm, and active learning is used to identify in the remaining corpus the examples that are "hard to tag". These are the examples that are presented to the contributors for tagging in *Stage 2*.

For all tagging, contributors are asked to select the sense (or senses) they find to be the most appropriate in a given sentence. The selection is made using check-boxes, containing all WordNet senses of the current word, plus two additional choices, "unclear" and "none of the above." The results of any automatic classification or the classification submitted by other users are not presented so as to not bias a contributor's decisions. Based on preliminary feedback from both researchers and contributors, the current version of the system allows contributors to specify more than one sense for a given instance.

2.1 *Source corpora*

Initially, the corpus for annotation was formed by sampling three different corpora, namely the *Penn Treebank* corpus, the *Los Angeles Times* collection as provided during TREC conferences (`http://trec.nist.gov`) and *Open Mind Common Sense* (`http://commonsense.media.mit.edu`), a collection of about 500,000 commonsense assertions in English as contributed by volunteers over the Web (Singh 2002). Recently, we have integrated the *British National Corpus*; we also plan to integrate the *American National Corpus* when it becomes available.

2.2 *Sense inventory*

The sense inventory used in the current system implementation is WordNet 1.7.1 (Miller 1995). Users are presented with the current sense definitions from WordNet, and asked to decide on the most appropriate sense(s) in the given context.

Future versions of the system may adopt a new sense inventory, or use the coarse sense classes derived from WordNet, since the fine granularity of WordNet was occasionally a source of confusion for some contributors and sometimes discouraged them from returning to the tagging task.

3 Quantity and quality of Web-based sense tagged corpora

Collecting from the general public holds the promise of providing much data at low cost. It also raises the importance of two aspects of data collection: (1) ensuring contribution quality, and (2) making the contribution process engaging to the contributors.

To ensure contribution quality, we collect redundant tagging for each item. The system currently uses the following rules in presenting items to volunteer contributors:

At least two tags per item. We present an item for tagging until agreement between at least two taggings has been obtained. Moreover, there is a maximum number of tags that we collect per item, which is currently set to four. That is, if no agreement is reached for the first four taggings, the item is dropped from the annotated corpus.[1]

One tag per item per contributor. We allow contributors to submit tagging either anonymously or having logged in. Anonymous contributors are not shown any items already tagged by contributors (anonymous or not) from the same IP address. A logged in contributor is not shown items that this contributor has already tagged.

In one year since the beginning of the activity, we collected more than 100,000 individual sense tags from contributors. Of these, approximately 16,500 tags came from using the system in the classrooms as a teaching aid (for which the web site provides special features). Future rate of collection depends on the site being listed in various directories and on the contributor repeat visit rate. We are also experimenting with prizes to encourage participation.

We measured the quality of the collected data in two ways. One is *inter-tagger agreement*, (including κ statistics), which measures agreement between the tags assigned to the same item by two different annotators. The other is *replicability*, which measures the degree to which an annotation experiment can be replicated. According to Kilgarriff (1999), the ability to recreate closely agreeing tagging by doing a "second run" is an even more telling indicator of annotation quality than inter-tagger agreement.

3.1 *Inter-tagger agreement*

The inter-tagger agreement obtained so far is closely comparable to the agreement figures previously reported in the literature. Kilgarriff (2001) mentions that for the SENSEVAL-2 nouns and adjectives there was a 66.5% agreement between the first two taggings (taken in order of submission) entered for each item. About 12% of that tagging consisted of multi-word expressions and proper

[1] Note that most figures reported in this chapter refer to an earlier version of the system, which was restricted to collect no more than two taggings per item.

nouns, which are usually not ambiguous, and which are not considered during our data collection process. So far we measured a 62.8% inter-tagger agreement between the first two taggings for single word tagging, plus close-to-100% precision in tagging multi-word expressions and proper nouns (as mentioned earlier, this represents about 12% of the annotated data). This results in an overall agreement of about 67.3% which is reasonable and closely comparable with previous figures.

3.2 *Kappa statistic*

In addition to raw inter-tagger agreement, we also calculated the kappa statistic, which removes from the agreement rate the amount of agreement that is expected by chance (Carletta 1996).

We measure two figures: *micro-average* κ, where number of senses, agreement by chance, and κ are determined as an average for all words in the set, and *macro-average* κ, where inter-tagger agreement, agreement by chance, and κ are individually determined for each of the 280 words in the set, and then combined in an overall average. With an average of five senses per word, the average value for the agreement by chance is measured at 0.20, resulting in a *micro*-κ statistic of 0.58. For *macro*-κ estimations, we assume that word senses follow the distribution observed in the Open Mind annotated data, and under this assumption, the *macro*-κ is 0.35.

Only few previous sense annotation experiments report on the κ statistic, and therefore it is hard to compare the values we obtain with previous evaluations. It is generally assumed that agreement above 0.80 represents *perfect agreement*, 0.60-0.80 represents *significant agreement*, 0.40-0.60 represents *moderate agreement*, and 0.20-0.40 is *fair agreement*. While most NLP applications seek data with an agreement that is at least *significant*, this is rarely the case in the task of sense annotation. Previous semantic annotation experiments report a *macro*-κ for nouns of 0.30 (Ng 1999), as measured on the intersection between SemCor and the DSO corpus; a value of 0.49 for the annotation of 36,000 word instances in a French corpus (Veronis 2000); a value of 0.44 for the Spanish SENSEVAL-2 task (Rigau 2001) (for the last two κ values, it is not clear whether they were computed using *micro* or *macro* average).

We also measure the κ statistic on the corpus constructed for the 29 nouns in the English lexical sample task at SENSEVAL-2. The SENSEVAL-2 English lexical sample data was constructed following the principle of *tag until at least two agree*. To create a setting similar to the Open Mind collection process, in this evaluation we only consider the first two tags (arranging tags in order of submission). With agreement by chance determined based on sense distributions drawn from the corpus itself, the *macro*-κ statistic for this sense tagged corpus is 0.62, and the *micro*-κ statistic is 0.65. On the same noun set, the Open Mind

data has a *macro-*κ value of 0.43, and a *micro-*κ of 0.55. While κ statistics for the SENSEVAL-2 data are clearly higher, the figures are not, however, directly comparable since (1) SENSEVAL-2 data also includes multi-word expressions, which are usually easy to identify, and lead to high agreement rates; and (2) in Open Mind Word Expert, the instances to be tagged for this set of 29 nouns were selected using an *active learning* process, and therefore these instances are "hard to tag".

Additionally, we performed an experiment where only clearly separable word senses were listed on the *Open Mind Word Expert* site for selection in tagging (we used six different senses for *line*, as used in (Leacock 1998)). In this case, $\kappa = 0.73$, a higher figure than the case where fine grained sense distinctions are used in tagging. This suggests that a sense inventory with clearer sense distinctions is likely to have a positive impact on the inter-tagger agreement.

3.3 *Replicability*

To measure the replicability of the tagging process performed by Web users, we carried out a tagging experiment for which annotation performed by "trusted humans" already existed. We used the data set for the noun *interest*, made available by Bruce and Wiebe 1994. Because this 2,369-item data set was originally annotated with respect to LDOCE, we had to map the sense entries from LDOCE to WordNet in order to make a direct comparison with the data we collect. The mapping was straightforward with one exception: all six LDOCE entries mapped one-to-one onto WordNet senses. There was one additional WordNet entry not defined in LDOCE; for this entry we discarded all corresponding examples from the Open Mind annotation.

Next, we identified and eliminated all the examples in the corpus that contained collocations (e.g., *interest rate*) because these collocations have unique WordNet senses. These examples accounted for more than 35% of the data. Finally, the remaining 1,438 examples were displayed on the Web-based interface for tagging.

Out of the 1,438 examples, 1,066 had two tags that agreed, therefore a 74% inter-annotator agreement for single words tagging.[2] Out of these 1,066 items, 967 had a tag that coincided with the tag assigned in the experiments reported in (Bruce 1994), which leads to an 90.8% replicability for single words tagging (note that the 35% monosemous multi-word expressions are not taken into account by this figure). This is close to the 95% replicability scores mentioned in (Kilgarriff 1999) for annotation experiments performed by lexicographers.

In all, robustness of our data is also corroborated by the experience of a similar volunteer contribution project (Singh 2002), which observed surprisingly

[2] Addition of the 35% monosemous multi-word expressions tagged with 100% precision leads to an overall 83% inter-tagger agreement for this particular word.

Word	Set size	Baseline	WSD	Word	Set size	Baseline	WSD
activity	103	90.00%	90.00%	arm	142	52.50%	80.62%
art	107	30.00%	63.53%	attitude	107	100.00%	100.00%
bank	160	91.88%	91.88%	bar	107	61.76%	70.59%
bed	142	98.12%	98.12%	blood	136	91.05%	91.05%
brother	101	95.45%	95.45%	building	114	87.33%	88.67%
captain	101	47.27%	48.18%	car	144	99.44%	99.44%
cell	126	89.44%	88.33%	chance	115	56.25%	81.88%
channel	103	84.62%	86.15%	chapter	137	68.50%	71.50%
child	105	55.33%	84.67%	circuit	197	31.92%	45.77%
coffee	115	95.00%	95.00%	day	192	34.76%	44.76%
degree	140	71.43%	82.14%	device	106	98.12%	98.12%
doctor	133	100.00%	100.00%	dog	130	100.00%	100.00%
door	112	54.62%	45.38%	eye	117	96.11%	96.11%
facility	205	81.60%	74.40%	father	160	96.88%	96.88%
function	105	24.67%	32.00%	god	110	71.82%	81.82%
grip	239	45.94%	61.88%	gun	143	94.71%	94.71%
hair	147	96.67%	96.67%	horse	138	100.00%	100.00%
image	120	36.67%	71.67%	individual	103	96.15%	96.15%
interest	1066	39.91%	71.08%	kid	106	83.75%	84.38%
law	106	38.12%	66.88%	letter	137	85.00%	81.00%
list	102	100.00%	100.00%	material	196	77.60%	76.40%
mother	119	99.00%	99.00%	mouth	151	74.38%	77.50%
name	136	98.42%	98.42%	object	183	96.19%	96.19%
office	209	62.76%	61.03%	officer	103	56.15%	55.38%
people	120	99.17%	99.17%	plant	126	98.89%	98.89%
pressure	106	72.50%	70.62%	product	216	80.74%	81.48%
report	101	66.36%	60.91%	rest	360	51.11%	67.22%
restraint	204	22.92%	46.25%	room	124	100.00%	100.00%
sea	205	90.80%	90.80%	season	102	92.50%	92.50%
song	116	92.35%	92.35%	structure	112	75.38%	72.31%
sun	101	63.64%	66.36%	term	125	71.18%	90.59%
treatment	108	67.78%	66.67%	tree	105	100.00%	100.00%
trial	109	87.37%	86.84%	type	135	92.78%	92.78%
unit	108	54.44%	46.67%	volume	103	63.85%	54.62%
water	103	53.85%	72.31%				

Table 1: *Words with more than 100 sense tagged examples: (1) set size, (2) precision attainable with the most frequent sense heuristic, (3) precision attainable with the WSD system*

low the rate of maliciously misleading or incorrect contributions.

4 Exploiting agreement of human annotators for WSD

We also carried out two sets of WSD experiments to further evaluate annotation quality. For these experiments, we used the items for which two Web annotators agreed on the sense tag assigned. One set of experiments disambiguated a held

out subset of the collected corpus, with evaluations performed using ten-fold cross validations runs. This is the *intra-corpus* experiment, were both training and test sets are from the same source. The second set of experiments involves *inter-corpora* evaluations, in which the training corpus provided for the SENSEVAL-2 evaluation exercise is augmented with the examples contributed by Web users, and the performance is subsequently tested on the SENSEVAL-2 test data.

4.1 *Intra-corpus WSD*

In this experiment, we employ STAFS, one of the best performing WSD systems at SENSEVAL (Mihalcea 2002). In current experiments, we use only a small set of features, consisting of the target word itself, its part of speech, and a surrounding context of two words and their corresponding parts of speech. The WSD performance is evaluated during 10-fold cross validation runs. We also compute a simple baseline which always assigns the most frequent sense (also computed during 10-fold cross validation runs). Table 1 lists: all words for which we collected sense tagged data with at least 100 annotated examples available; the number of items with full inter-annotator agreement; the most frequent sense baseline; the precision achieved with STAFS.
For the total of 280 words for which data was collected, an average of 87 examples per word were annotated.[3] The most frequent sense heuristic yields correct results in 63.32% overall. When disambiguation is performed using STAFS, with a simple set of features consisting of the word itself, the word's part of speech, and a surrounding context of two (words and their corresponding parts of speech), the overall precision is 66.23%, which represents an error reduction of about 9% with respect to the most frequent sense heuristic.

Number of training examples	Precision		Error rate reduction
	baseline	STAFS	
any	63.32%	66.23%	9%
> 100	75.88%	80.32%	19%
> 200	63.48%	72.18%	24%
> 300	45.51%	69.15%	43%

Table 2: *Precision and error rate reduction for various sizes of the training corpus*

Moreover, the average for the 72 words which have at least 100 training examples (the words listed in Table 1) is 75.88% for the most frequent sense heuristic, and 80.32% when using STAFS, resulting in an error reduction of 19%. When at least 200 examples are available per word, the most frequent sense

[3] The Open Mind Word Expert sense tagged corpora used in these experiments is free for download at http://teach-computers.org

	S-2		S-2 + OMWE			S-2		S-2 + OMWE	
Word	Fine	Coarse	Fine	Coarse	Word	Fine	Coarse	Fine	Coarse
art	60.2%	65.3%	61.2%	68.4%	authority	70.7%	85.9%	76.1%	90.2%
bar	45.7%	58.3%	46.4%	57.0%	bum	75.6%	77.8%	66.7%	78.9%
chair	81.2%	81.2%	82.6%	82.6%	channel	52.1%	54.8%	49.2%	56.2%
child	60.9%	62.5%	56.2%	57.8%	church	62.5%	62.5%	67.2%	67.2%
circuit	49.4%	49.4%	48.2%	50.6%	day	65.5%	65.5%	66.2%	67.6%
detention	68.8%	68.8%	71.9%	71.9%	dyke	82.1%	82.1%	82.1%	82.1%
facility	58.6%	93.1%	48.3%	94.8%	fatigue	79.1%	83.7%	76.7%	81.4%
feeling	64.7%	64.7%	64.7%	64.7%	grip	54.7%	74.5%	62.7%	70.6%
hearth	71.9%	87.5%	71.9%	87.5%	holiday	77.4%	83.9%	77.4%	87.1%
lady	77.4%	86.8%	77.4%	88.7%	material	50.7%	60.9%	53.6%	66.7%
mouth	56.7%	85.0%	66.7%	90.0%	nation	70.3%	73.0%	70.3%	73.0%
nature	65.2%	76.1%	69.6%	84.8%	post	58.2%	62.0%	57.0%	60.8%
restraint	48.9%	60.0%	57.8%	68.9%	sense	54.7%	54.7%	58.5%	60.4%
spade	57.6%	57.6%	51.5%	51.5%	stress	56.4%	82.1%	56.4%	82.1%
yew	78.6%	96.4%	78.6%	96.4%	Average	63.99%	72.27%	64.58%	73.78%

Table 3: *Evaluation using Senseval-2 (S-2) and Web-users (OMWE) examples*

heuristic is correct 63.48% of the time, and the WSD system is correct 72.18% of the time, which represents a 24% reduction in disambiguation error. See Table 2 for precision and error rate reduction for various sizes of the training corpus.

For the words for which more data was collected from Web users, the improvement over the most frequent sense baseline was larger. This agrees with prior work by other researchers (Ng 1997), who noted that additional annotated data is likely to bring significant improvements in disambiguation quality.

4.2 *Inter-corpora experiments*

In these experiments, we enlarge the set of training examples provided within the Senseval evaluation exercise with the examples collected from Web users, and evaluate the impact of the additional training examples on performance. Table 3 shows the results obtained on the test data when only SENSEVAL-2 training data were used, and the results obtained with both SENSEVAL-2 and Web-users training examples. The same WSD system is used in this experiment. Only examples pertaining to single words are used (that is, we eliminate the SENSEVAL-2 examples pertaining to collocations).

There is a small error rate reduction of 2% for fine grained scoring.[4] A more significant error reduction of 5.7% was observed for coarse grained scoring. Notice that the examples used in our Web-based system are drawn from a corpus completely different than the corpus used for SENSEVAL-2 examples, and therefore the sense distributions are usually different, and often do not match the test

[4] Fine grained scoring is a performance evaluation using word senses as defined in WordNet. Coarse grained scoring is an evaluation that relies on similar senses being grouped in clusters (e.g., by lexicographers).

data sense distributions (as is the case when train and test data are drawn from the same source). Previous word sense disambiguation experiments performed across diverse corpora have shown that variations in genre and topic negatively affect performance (Martinez 2000). The relatively low error reductions obtained in our own inter-corpora experiments confirm these results.

5 Summary

We proposed a solution for construction of large sense tagged corpora with volunteer contributions over the Web. We evaluated the quantity and quality of the data collected from Web users, and showed how these data can be used to improve WSD performance. The experiments performed with these data showed that the inter-tagger agreement, replicability, and disambiguation results obtained on these data are comparable with what can be obtained using data collected with the traditional method of hiring lexicographers, at a much lower cost.

REFERENCES

Bruce, Rebecca & Janyce Wiebe. 1994. "Word Sense Disambiguation Using Decomposable Models". *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics* (*ACL'94*), 139–146. Las Cruces, New Mexico.

Carletta, Jean. 1996. "Assessing Agreement on Classification Tasks: The Kappa Statistic". *Computational Linguistics* 22.2:249–254.

Kilgarriff, Adam. 1999. "95% Replicability for Manual Word Sense Tagging". *Proceedings of European Association for Computational Linguistics* (*EACL'99*), 277–278. Bergen, Norway.

Kilgarriff, Adam. 2001. "English Lexical Sample Task Description". *Proceedings of the Senseval-2 Workshop at the 39th Annual Meeting of the Association for Computational Linguistics*, 17–20. Toulouse, France.

Leacock, Claudia & Martin Chodorow & George Miller. 1998. "Using Corpus Statistics and WordNet Relations for Sense Identification", *Computational Linguistics* 24.1:147-165.

Martinez, David & Eneko Agirre. 2000. "One Sense per Collocation and Genre/ Topic Variations". *Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora* (*EMNLP-2000*), 207–215. Hong Kong.

Miller, George. 1995. "WordNet: A Lexical Database". *Communications of the ACM* 38.11:39–41.

Mihalcea, Rada. 2002. "Instance Based Learning with Automatic Feature Selection Applied to Word Sense Disambiguation". *Proceedings of the 19th International Conference on Computational Linguistics* (*COLING'02*), 660-666. Taipei, Taiwan.

Ng, Hwee Tou. 1997. "Getting Serious about Word Sense Disambiguation". *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, 1–7. Washington, D.C.

Ng, Hwee Tou & Chung Yong Lim & Shou King Foo. 1999. "A Case Study on Inter-Annotator Agreement for Word Sense Disambiguation". *ACL Workshop on Standardizing Lexical Resources* (*SIGLEX-99*), 9–13. College Park, Maryland.

Rigau, German, Mariona Taule & Ana Fernandez & Julio Gonzalo. 2001. "Framework and Results for the Spanish SENSEVAL". *SENSEVAL-2 Workshop; 39th Annual Meeting of the Association for Computational Linguistics*, 41–44. Toulouse, France.

Singh, Push. 2002. "The Open Mind Common Sense Project". — `http://www.kurzweilai.net/` [Source checked in May 2004]

Veronis, Jean. 2000. "Sense Tagging: Don't Look for the Meaning but for the Use". *Computational Lexicography and Multimedia Dictionaries* (*COMLEX'2000*), 1–9. Kato Achia, Greece.

Reducing False Positives by Expert Combination in Automatic Keyword Indexing

ANETTE HULTH
Stockholm University

Abstract

This work extends previous work on automatic keyword indexing by showing how the number of incorrectly assigned keywords — as measured by keywords assigned by professional indexers — may be highly reduced by combining the predictions of several classifiers. The classifiers are trained on different representations of the data, where the difference lies in the definition on what constitutes a candidate term in a written document.

1 Introduction

The work described in this chapter concerns automatic keyword indexing,where the goal is to automatically find a set of terms describing the content of a single document. The motivation for the work is that keywords can play an important role in supporting a user in the search for relevant information. The keywords may serve as a dense summary for a document, they can lead to an improved performance of a search engine, or they may constitute the input to a clustering mechanism, to give just some examples. Most documents lack keywords, and since it would be much too time-consuming, expensive, as well as tedious to manually assign keywords for all these documents, automatic methods must be explored.

The approach taken to automate the task is that of supervised machine learning, that is, a prediction model (or a classifier) is constructed by training a learning algorithm on documents with known keywords. The model is subsequently applied to previously unseen documents, to select a suitable set of keywords. This approach to automatically assign keywords is also used by for example, Frank et al. (1999), Turney (2000), and Pouliquen et al. (2003). More specifically, the work presented here concerns keyword *extraction*, where the selected keywords are present verbatim in the document to which they are assigned. To evaluate the constructed models, manually assigned keywords are used as the gold standard.

Automatic keyword indexing is a difficult task, and the performance of the state-of-the-art keyword extraction is much lower than for many other NLP tasks, such as parsing and tagging. The low performance is partly due to the chosen evaluation method: It is to a certain extent inevitable that keywords selected by a classifier differ from those selected by a human indexer, as not even professional indexers agree on what set of keywords best describes a document. One reason

for this disagreement is that keyword indexing deals with natural language, and humans are often inconsistent in such tasks. This, in turn, is because human languages allow for syntactic and semantic variations.

The work presented here builds on work by Hulth (2003), in which several classifiers for automatic keyword extraction were evaluated. The most evident drawback with the evaluated classifiers was that they selected too many keywords that were not chosen as keywords by the human indexers. In some cases it may be desirable to have a prediction model that assigns more keywords than a human would do, for example if the classifier is to be used for semi-automatic indexing. In that case the goal for the training should be to find all manually assigned keywords, as well as a limited set of additional ones. The final selection would then be made by a professional indexer. Semi-automatic indexing was, however, not the purpose of the experiments described in Hulth (2003).

In this chapter, experiments on how the number of incorrectly assigned keywords was reduced to a more acceptable level — as measured by keywords assigned by professional indexers — are described. The improvement was obtained by taking the majority vote of three classifiers which each was trained on a different representation of the data. The representations differed in how the candidate terms were selected from the documents; using different definitions on what constitutes a term in a written text.

The outline is as follows: In the next section, a summary of the classifiers used for the expert combination is given. In Section 3, the ensemble technique and its results are described. Also, two approaches that did not work in reducing the amount of false positives are shortly presented. (A false positive is a candidate term that has been given the label *positive* by the classifier although its true value is *negative*, that is, it is not a manually assigned keyword.) Before concluding and giving some directions for future work, an example is given of the automatic keyword extraction before and after the expert combination.

2 Training the classifiers

One of the most important aspects in machine learning is how the data are preprocessed and represented,and consequently what is given as input to the learning algorithm. In the case of automatic keyword extraction, the pre-processing consists of two steps. The first step is to determine where a keyword begins and ends in a running text, since a keyword may consist of one or several tokens. Whereas humans usually are good at knowing when a number of consecutive tokens should be treated as a unit (as one "word" or as one phrase), this is not evident for an automatic indexer. This procedure is referred to as defining the *candidate terms*. Once the candidate terms have been extracted, it is likely that too many terms have been selected to make them useful as keywords. A filtering of terms is achieved by constructing a model that — based on the values of a num-

ber of defined features — can make this distinction for the extracted candidate terms. The second step is thus to calculate the feature values for the candidate terms. The goal for the learning is then to find the feature values discriminating the candidate terms that are appropriate keywords from those candidate terms that are inappropriate. To train the prediction models, the feature values of the known keywords are used.

In the experiments on automatic keyword extraction discussed in Hulth (2003), three different approaches to select the candidate terms from the documents were used. These were all stemmed:

- uni-, bi-, and trigrams excluding stopwords (referred to as n-*grams*)
- noun phrase (NP) chunks
- words matching any of a set of empirically defined part-of-speech (POS) tag sequences, where the patterns corresponded to frequently occurring patterns of manual keywords (referred to as *patterns*).

Three features were selected for the candidate terms. These were:

- term frequency
- collection frequency (IDF)
- relative position of the first occurrence.

In addition, experiments with a fourth feature — that significantly improved the results — were performed for each term selection approach. This feature was:

- the most frequent POS tag sequence assigned to the candidate term.

In total, six models were evaluated on the test set (three term selection approaches with three and four features respectively). The measures used for the evaluation were *precision* (how many of the automatically assigned keywords that are also manually assigned keywords), *recall* (how many of the manually assigned keywords that are found by the automatic indexer), and *F-measure* ($F_{\beta=1}$) for the selected keywords. To calculate the recall, the number of manually assigned keywords actually present in the abstract to which they were assigned was used. A keyword was considered correct if its stemmed form was equivalent to a stemmed manually assigned keyword.

The machine learning method used in the experiments was an implementation of *rule induction*, where the prediction models constructed from the given examples consist of a set of rules. The system applied is called *Rule Discovery System* (RDS 2003). The strategy used to construct the rules was *recursive partitioning* (or *divide-and-conquer*), which has as its goal to maximise the separation of the classes for each rule. The resulting rules are hierarchically organised (that is, as decision trees). In order to construct rules that are not over-fitted — to avoid rules that are too closely tuned to the training data and will generalise poorly — half of the training data are saved for pruning the rules. This action is part of how the learning algorithm is implemented in RDS. The results from

Hulth (2003) that are relevant for the experiments discussed in this chapter are found in Table 1. The presented results are for four features. For each term selection approach is shown: The number of assigned keywords in total (Assign.); the number of correct keywords in total (Corr.); the precision; the recall; and the F-measure. There are 500 documents in the test set. The total number of manually assigned keywords present in the abstract is 3 816, and the mean is 7.63 keywords per document.

Method	**Assign.**	**Corr.**	**Prec.**	**Recall**	**F-measure**
n-grams	7 815	1 973	25.2	51.7	33.9
NP-chunks	4 788	1 421	29.7	37.2	33.0
Patterns	7 012	1 523	21.7	39.9	28.1

Table 1: *Results for the individual classifiers*

In the experiments described here, the same data were used as in the previous experiments: A set of 2 000 abstracts in English from scientific journal papers with keywords assigned by professional indexers. Also, the division of the training (1 000 documents; to construct the prediction models), validation (500 documents; to evaluate the models, and select the best performing one), and test (500 documents; to get unbiased results) sets was kept.

3 Combining the experts

It has often been shown that combining experts leads to an improved accuracy, and there are several ways to apply ensemble techniques (see for example, Dietterich (1998)). Basically, one may either manipulate the training data; for example in both *bagging* and *boosting* a number of classifiers are obtained by training on different subsets of the whole data set. Or, one may use different learning algorithms on the same training data to acquire different classifiers. There is also the question of how to combine the classifiers to consider, for example whether better performing classifiers should be given higher weights in the ensemble.

3.1 *Combining different representations*

In this section, an ensemble method that highly reduced the number of incorrectly assigned keywords, while still retaining a large number of correct keywords, will be described. The ensemble was built from classifiers trained on different representations of the data. As mentioned in Section 2, six models were evaluated on the test set in Hulth (2003): Three term selection approaches with two sets of features (with or without the POS tag feature).

To reduce the number of incorrect keywords a pair-wise combination was initially made over the six models, given that the term selection approaches were different. Thus, in total twelve pairs were obtained. In order to be considered a keyword, a term had to be selected by both prediction models in the pair, that is, an unanimity vote was used.

The twelve pairs were evaluated on the validation data, and the three pairs (one pair for each combination of the term selection approaches) with the highest precision were selected. The reason for choosing precision as the selection criteria was that the goal is to reduce the false positives, thus the proportion of these should be as small as possible.

The three pairs that were selected were:

- *n*-grams with the POS tag feature + NP-chunks with the POS tag feature
- *n*-grams with the POS tag feature + patterns with the POS tag feature
- NP-chunks with the POS tag feature + patterns with the POS tag feature,

in other words, the three term selection approaches with the POS tag feature. In Table 2, the results for these three combinations on the test set are shown. In the table, the number of assigned keywords in total; the number of correct keywords in total; the precision; the recall; and the F-measure are displayed. (The total number of manually assigned keywords present in the abstract in the test data is 3 816, and the mean is 7.63 keywords per document.) As can be seen in the table, *n*-grams + NP-chunks assign the set of 500 abstracts in total 2 004 keywords, that is, on average 4.01 keywords per document. Of these are on average 1.80 correct. If looking at the actual number of keywords assigned, 27 documents have 0, while the maximum number of keywords assigned is 21. The median is 4. For the *n*-grams + patterns pair, in total 3 822 keywords are assigned. Of the 7.64 keywords on average per document, 2.14 are correct. If examining the actual distribution, 8 documents have no keywords assigned, the maximum is 34, and the median is 7. Finally, the NP-chunks + patterns assign in total 1 684 keywords, that is, on average 3.37 keywords per document; of these are 1.42 correct. The maximum number of keywords actually assigned is 14. 32 documents have 0, and the median is 3 keywords per document.

Method pair	**Assign.**	**Corr.**	**Prec.**	**Recall**	**F-measure**
n-gram+NP-chunk	2 004	902	45.0	23.6	31.0
n-gram+Pattern	3 822	1 069	28.0	28.0	28.0
NP-chunk+Pattern	1 684	708	42.0	18.6	25.7

Table 2: *Results for the three best performing pairs*

As can be seen in this table, the precision has increased for all pairs, compared to the performance of the individual classifiers (see Table 1). However, the F-measure has decreased for all three combinations. As it is important not only

to assign correct keywords, but also to actually find the manually assigned keywords, recall is considered equally important (the reason for when calculating the F-measure giving β the value 1).

As these results were still not satisfactory from the point of view of the F-measure, an experiment with joining the results of the three pairs was performed, that is, by taking the union of the keywords assigned by the pairs. Doing this is in fact equivalent to taking the majority vote of the three individual classifiers in the first place. The results on the test set when applying the majority vote are found in Table 3 (Sub. not removed). In total, 5 352 keywords are then assigned to the test set. On average per document, 3.31 keywords are correct, and 7.39 are incorrect. The actual number of keywords varies between 41 and 0 for four documents, and the median is 10 keywords per document. The F-measure when taking the majority vote for the three classifiers is 36.1, thus higher than for any of the individual classifiers. The precision is also higher than for any of the component models, and the recall is higher than for two of the individual classifiers. If comparing this result to the *n*-gram approach, that has the highest F-measure, the number of false positives has decreased by 2 145, while 318 true positives are lost. The result of the majority vote may be compared to the number of keywords assigned to the test set by the professional indexers, where three documents have no keywords present in its abstract. The median is 7 keywords, and the maximum is 27, and there are in total 3 816 manually assigned keywords.

Majority vote	**Assign.**	**Corr.**	**Prec.**	**Recall**	**F-measure**
Sub. not removed	5 352	1 655	30.9	43.4	36.1
Sub. removed	4 369	1 558	35.7	40.8	38.1

Table 3: *Results for the majority vote*

As another improvement, the subsumed keywords may be removed, that is, if a term is a substring of another assigned keyword, the substring is removed. In Table 3 (Sub. removed) one can see that although some correctly assigned keywords are removed as well (5.9%), the number of false positives decreases by 24.0%. If looking at the actual distribution on the test set, four documents have 0 keywords. The maximum number of keywords assigned is 30, while the median is 8 keywords per document. This results in the highest F-measure (38.1) obtained on the test set for these experiments.

3.2 *Lessons learned*

Before any successful results were obtained on the task of reducing the number of incorrectly assigned keywords, two other methods were examined. In these two

experiments, combinations of classifiers were made for each term selection approach separately. In the first experiment, *bagging* (Breiman 1996) was applied, that is, from a training set consisting of *n* examples, a new set of the same size is constructed by randomly drawing *n* examples with replacement. (An *example* is a feature value vector for, in this case, each candidate term.) This procedure is repeated *m* times to create *m* classifiers. Both voting, with varying numbers of classifiers that should agree, as well as setting varying threshold values for the number of maximum keywords to assign to each document in combination with voting, was tried.

In the second unsuccessful experiment, the fact that the data set is unbalanced was exploited. By varying the weights given to the positive examples for each run, a set of classifiers was obtained. Thereafter voting was applied in the same fashion as for the first unsuccessful experiment. In addition, a simple weighting scheme was applied, where higher weights were given to classifiers that found more correct keywords.

As the results for these experiments were poor, they are not presented here. Although the number of false positives did decrease, too many of the true positives also disappeared. As these experiments did not succeed, it may be suspected that the selected features are not enough to discriminate keywords from non-keywords, at least not in this collection.

3.3 *An example of automatically assigned keywords*

An example of the automatic extraction will now be given. In Figure 1, an abstract from the test set is shown together with both the manually assigned keywords, as well as with the automatically assigned keywords using the majority vote, with the subsumed keywords removed. In Figure 2, the keywords selected by each individual classifier for this abstract are shown, while the keywords for the three pairs are found in Figure 3. In the figures, the terms in bold are also manually assigned keywords.

4 Conclusions and future work

The experiments and the evaluation presented in this chapter concerns automatic keyword extraction. I have here shown how the number of automatically assigned keywords that are incorrect may be highly reduced, while the number of correct keywords is still satisfactory, as measured by keywords assigned by professional indexers. The improvement is achieved by taking the majority vote for three classifiers, each trained on a different representation of the data. The representations differ in how the terms are selected from the documents. The three methods used are uni-, bi-, and trigrams; NP-chunks; and terms matching any of a set of POS patterns (see Hulth (2003) for details). If precision for some reason

ABSTRACT
Lung metastasis detection and visualization on CT images: a knowledge-based method. A solution to the problem of lung metastasis detection on computed tomography (CT) scans of the thorax is presented. A knowledge-based top-down approach for image interpretation is used. The method is inspired by the manner in which a radiologist and radiotherapist interpret CT images before radiotherapy is planned. A two-dimensional followed by a three-dimensional analysis is performed. The algorithm first detects the thorax contour, the lungs and the ribs, which further help the detection of metastases. Thus, two types of tumors are detected: nodules and metastases located at the lung extremities. A method to visualize the anatomical structures segmented is also presented. The system was tested on 20 patients (988 total images) from the Oncology Department of La Chaux-de-Fonds Hospital and the results show that the method is reliable as a computer-aided diagnostic tool for clinical purpose in an oncology department.

AUTOMATICALLY ASSIGNED KEYWORDS
clinical purpose; computed tomography; computer-aided diagnostic tool; ct images; image interpretation; knowledge-based top-down; la chaux-de -fonds hospital; lung metastasis detection; oncology; radiotherapist; three-dimensional analysis; top-down approach; total images

MANUALLY ASSIGNED KEYWORDS
computed tomography; computer-aided diagnostic tool; ct images; image interpretation; knowledge-based top-down approach; lung metastasis detection; oncology; thorax; three-dimensional analysis

Figure 1: *An abstract with automatically and manually assigned keywords*

is considered more important than the F-measure — if the assigned keywords must have a high quality — a combination of either *n*-grams and NP-chunks or NP-chunks and patterns using an unanimity vote should be used instead.

In order to establish which errors that are specific for one term selection approach, while not present in the two other (that is, which types of errors are avoided by taking the majority vote), all false positives from ten arbitrarily selected documents in the validation set were collected. Unfortunately, it was difficult to categorise the errors made by each approach (as may be suspected when looking at Figure 2). One of the few things that could be noted was that some of the terms selected by the NP-chunk classifier began with a determiner (for example, 'a given grammar'). These terms are rarely keywords, and are not extracted by the two other approaches (most determiners are stopwords, and are not part of the POS patterns). However, a more thorough investigation must be

N-GRAMS
chaux-de-fonds; clinical purpose; **computed tomography**; **computer-aided diagnostic tool**; **ct images**; knowledge-based method; knowledge-based top-down; **knowledge-based top-down approach**; la chaux-de-fonds hospital; lung extremities; lung metastasis; metastases; metastasis detection; **oncology**; oncology department; ribs; **three-dimensional analysis**; top-down approach; total images

NP-CHUNKS
988 total images; clinical purpose; **computed tomography**; ct; **ct images**; **image interpretation**; la chaux-de-fonds hospital; **lung metastasis detection**; radiotherapist

PATTERNS
chaux-de-fonds hospital; **computer-aided diagnostic tool**; department; **image interpretation**; knowledge-based top-down; la chaux-de-fonds; lung metastasis; **lung metastasis detection**; **oncology**; radiotherapist; thorax contour; **three-dimensional analysis**; top-down approach; total images; tumors

Figure 2: *Assigned keywords for each term selection approach before the expert combination*

made, where also the selected keywords for each classifier are compared to all candidate terms extracted from the documents before the classifier is applied, to first establish which terms that are filtered out already on the classification level.

N-GRAMS + NP-CHUNKS
clinical purpose; **computed tomography**; **ct images**; la chaux-de-fonds hospital

N-GRAMS + PATTERNS
computer-aided diagnostic tool; knowledge-based top down; lung metastasis; **oncology**; **three-dimensional analysis**; top-down approach; total images

NP-CHUNKS + PATTERNS
image interpretation; **lung metastasis detection**; radiotherapist

Figure 3: *Assigned keywords for each pair of the term selection approaches*

An alternative to taking the majority vote, could be to rank the keywords according to how many of the individual classifiers that agree upon a candidate term being a keyword, thus obtaining a probabilistic output. First the classifiers would need to be ranked internally according to their previous performance. A threshold value for the maximum number of keywords to assign to each document would then have to be set, either by the system or by a user.

When inspecting the automatically assigned keywords, it may be concluded that using keywords assigned by one professional indexer as the gold standard for the evaluation does not always give justification to the models. Many automatically selected keywords have a clear relation to the subject at hand, although not chosen by the human for one reason or another. Alternatives to this type of evaluation are currently under investigation.

Acknowledgements. For valuable comments and suggestions: Beáta Megyesi and Tony Lindgren.

REFERENCES

Breiman, Leo. 1996. "Bagging Predictors". *Machine Learning* 24:2.123-140.

Dietterich, Thomas G. 1998. "Machine Learning Research: Four Current Directions". *The AI Magazine* 18:4.97-136.

Frank, Eibe, Gordon W. Paynter, Ian H. Witten, Carl Gutwin & Craig G. Nevill-Manning. 1999. "Domain-Specific Keyphrase Extraction". *Proceedings of the International Joint Conference on Artificial Intelligence* (*IJCAI'99*), 668-673. Stockholm, Sweden.

Hulth, Anette. 2003. "Improved Automatic Keyword Extraction Given More Linguistic Knowledge". *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (*EMNLP 2003*), 216-223. Sapporo, Japan.

Pouliquen, Bruno, Ralf Steinberger & Camelia Ignat. 2003. "Automatic Annotation of Multilingual Text Collections with a Conceptual Thesaurus". *Proceedings of the Workshop on Ontologies and Information Extraction held as part of the EUROLAN-2003 summer school on The Semantic Web and Language Technology — Its Potential and Practicalities*, 19-28. Bucharest, Romania.

RDS. 2003. Rule Discovery System. `www.compumine.com`. Stockholm: Compumine AB.

Turney, Peter D. 2000. "Learning Algorithms for Keyphrase Extraction". *Information Retrieval* 2:4.303-336.

Socrates: A Question Answering Prototype for Bulgarian

HRISTO T. TANEV

ITC-irst, Trento

Abstract

In this paper we present Socrates - a prototype of a QA system for the Bulgarian language. The system exploits both encyclopaedic resources and information found on the web. The approach uses linguistic patterns and indicators to rank the results obtained from the web. An encyclopaedic virtual database VIDA was created to provide uniform interface to different on-line and offline encyclopaedic resources. The system was tested on 100 questions from TREC 2001 translated in Bulgarian. The system answered correctly 52 of them.

1 Introduction

Open domain Question Answering (QA) is a research area which aims at effective information retrieval rather than just document retrieval (Hirschman 2001). The goal of an open domain QA system is to identify the answer to a domain-independent question either on the web (Bucholz 2001) or in a local document collection (Voorhees 2001). QA systems are presented with natural language questions and the expected output is either the actual answer identified in a text or small text fragments containing the answer.

Under the promotion of the TREC competitions, Open Domain Question Answering has emerged as a new research topic in the field of Information Retrieval (Voorhees 2001). However, current approaches for Information Extraction and Question Answering (QA) are oriented mainly toward the English language. This contrasts with the situation on the web where a great number of languages are represented. People speaking different languages access the Internet millions of times every day, searching for information. Very often specific information is available only in some other language than English.

Multilingual QA was introduced as a new task in the CLEF 2003 (Cross Language Evaluation Forum) (`http://clef-qa.itc.it`); QA tasks for different European languages were promoted in CLEF: English, French, Dutch, Italian and Spanish. However, many languages, among them Bulgarian, remain out of the focus of such forums, although there is a significant amount of information on the web and other machine readable sources written in these languages. No previous work has addressed the problems of open domain Question Answering from sources in Bulgarian and other Slavonic languages.

This paper presents work in progress in the area of open domain QA for Bulgarian. A prototype of a QA system was built which answers to three types of questions: definition questions, Where-Is questions and temporal questions. The system was tested on 100 questions from TREC 2001, translated in Bulgarian; the system answered correctly 52 of them.

2 Overview

Socrates is a prototype of an open domain QA system for Bulgarian which uses both encyclopaedic resources and information on the web to find answers to questions in the Bulgarian language. The use of heterogeneous information sources which have different levels of coverage and structure are important for high precision QA. For example, the search for the date of the beginning of the World War II in an encyclopaedia will be relatively easy since only the entry about the World War II should be analysed, while the search on the web for the same information will return a lot of documents which tell about the war. However, when the answer is not found in the encyclopaedia a QA system may find relevant documents on the web, which has broader coverage.

Socrates combines pattern based techniques for question answering (Ravichandran & Hovy 2002, Soubbotin & Soubbotin 2002), linguistic analysis and vector search IR from pre-processed encyclopaedic database. The LINGUA language engine for Bulgarian (Tanev & Mitkov 2002) is in the core of the linguistic processing in Socrates.

The system does not perform named entity recognition, nor does it use any semantic information. The core of the approach lies in applying manually created linguistic patterns on text retrieved from the web and use of encyclopaedic resources.

2.1 *Question types*

Currently, the prototype is able to answer three classes of questions:

Definition questions, which ask for a definition of a noun, noun phrase or a person name. Examples for definition questions are “Kakvo e kofein?” (“What is caffeine?”) , “Koy e Galileo?” (“Who is Galileo?”). Each question of this type has a *focus* - a word or a phrase for which we search for a definition. For example, “kofein”(“caffeine”) and “Galileo” are the focuses of the two questions shown hereabove. In the TREC 2001 QA track about 20% of the questions were definition questions. The organisers of the QA track explained this fact with the large number of definition questions in the FAQ lists on the web.

Where-Is questions. These questions are built according to the pattern: *Kade e* ⟨*NP*⟩*?* (*Where is* ⟨*NP*⟩*?*) where ⟨NP⟩ stands for a noun phrase. Such

a question type is used when the user searches for the location of a city, island, state, landmark, etc.

Temporal questions. This class of questions ask for a date of event. In contrast with the Where-Is questions Socrates accepts temporal questions without any restriction on the syntactic form. An example of temporal question is the question: "Koga e potanal Titanik?" ("When did the Titanic sink?")

2.2 *The system architecture*

A traditional QA system uses the following modules:

1. Question processing and question type identification
2. Document retrieval
3. Document processing
4. Answer extraction and ranking from the retrieved documents

Question processing. The user chooses from a menu the question stem, thus defining the type of the question. For example, to ask for a definition question, the item "Definition questions" is selected from the main menu and then the user can choose between one of the question stems: "Kakvo e" ("What is"), "Koy e" ("Who is") or "Koya e" ("Who is") (feminine); then, the rest of the question is entered. For the Definition questions Socrates generates all the possible morphological forms of the word or phrase which is the focus of the question. For the temporal questions part-of-speech tagging is performed to identify the content words.

Document retrieval. Socrates performs information retrieval from two sources: the virtual encyclopaedic database VIDA and the web. Document retrieval has two stages: First, Socrates searches VIDA entries which may answer the question and then a query is sent to the Google API (`http://www.google.com/apis`). Socrates takes the document snippets returned by the search engine, it also downloads the top-ranked documents.

Document processing. Socrates performs sentence splitting, part-of-speech tagging and noun phrase extraction on the results returned from Internet. The language engine LINGUA (Tanev & Mitkov 2002) is used to perform this processing. Socrates performs pattern matching on the linguistic structures built by the language engine in order to score and rank them with respect to their relevance.

Answer extraction and ranking. Socrates performs sentence level answer extraction from the Google snippets and documents. When the source of the answer is a Google snippet, the system returns a text fragment, not the whole sentence.

If an entry is found in VIDA which may contain the answer, the whole entry is returned, because the VIDA entries are compact in size and usually no further restriction of the context is necessary.

Answer ranking takes into account the source of the answers and the presence of linguistic patterns in the answer context. If results are found in the encyclopaedic database VIDA they are ranked first, since the search in this resource is more reliable than the same search on the web. Next, the results from the web are ordered according to the score obtained from a pattern matching procedure. A set of linguistic and statistical clues, called indicators, is used together with the patterns to rank the results from Internet.

Socrates returns a a list of sentences or fragments containing the answer and also links to the documents from which the answers have been extracted. For the definition questions the user can also choose a set of web addresses returned from the system, for which Socrates scans the pages and extracts all the text contexts in which the definition focus appears.

3 Pattern-based answer extraction

The system uses manually created linguistic patterns to retrieve from the web fragments containing candidate answers. The patternswere created after studying morpho-syntactic structures which appear in the answers to different question types (currently location and definition questions). After applying the patterns, the extracted answering fragments are sorted using a system of rules for quantitative evaluation.

Patterns are used for two types of questions: definition questions and "Kade e" ("Where is") questions. Regarding the definition questions, some studies point out that the pattern approach is very appropriate for these types of questions (Ravichandran & Hovy 2002). Socrates makes use of linguistic patterns and rules, which consider syntactic, morphological and lexical information in the answer context. The system also considers the frequencies of the elements which appear in the patterns. As no semantic resources are available for Bulgarian and no named entity recognition is performed by the system, Socrates relies entirely on syntactic, statistical clues, and a list of important definition words to extract and rank the text fragments containing the answer.

Socrates extracts answer-containing fragments by matching the processed documents with a set of patterns relevant to the question type. Then, a set of linguistic indicators are activated for every candidate. These indicators give a positive or negative score to every candidate fragment. The score from all the indicators are summed up and the sentences are accepted or rejected on the basis of that score. For example one of the definition question indicators "auxiliary verb definition" which gives a positive score is activated when all of the following conditions hold:

(1) The pattern is *X e* ⟨*NP*⟩ (*X is* ⟨*NP*⟩) or *X sa* ⟨*NP*⟩ (*X are* ⟨*NP*⟩);
(2) X has male gender and ⟨NP⟩ has male gender, or X has neutral or female gender;

(3) both X and ⟨NP⟩ are singular or plural and their number coincides with the number of the auxiliary verb;
(4) no other noun phrase includes in itself ⟨NP⟩; and
(5) there is no preposition before X.

This indicator is activated when a typical definition is found: "slanchevite petna sa oblasti vav fotosferata s niska temperatura" ("The sunspots are low temperature zones in the photosphere"). If we search for the answer to the question "Kakvo e lunata?" ("What is the Moon?") the indicator will be activated for phrases like: "Lunata e edinstveniat estestven spatnik na zemiata" ("The Moon is the only natural satellite of the Earth"), but will not be activated when the following phrase is encountered: "Orbitata na lunata e elipsa" ("The orbit of the Moon is an ellipse") In this case constraint 5 will impede the indicator from activating.

For answering definition questions for persons (e.g., "Who is Galileo?") we used an additional resource - a word list in which three groups of words are present: verbs, nouns and adjectives which tend to appear frequently in definition fragments. All these words are used when famous people and events are described. For example words like "medal", "nagrada"("prize"), "champion" are used when describing some famous sportsman. We created manually this list of words after studying the lexical peculiarities of the definitions in Bulgarian language.

A statistical indicator is applied when fragments matching Where-Is patterns are evaluated. This indicator returns the frequency with which a capitalised word appears as a location in fragments which match Where-Is patterns.

For example if the question is: "Kade e Sofia?" ("Where is Sofia?") Given these fragments: "... konferenciata v Sofia, Balgaria, ...", "... Posetete Sofia v Balgaria ...", "... Sofia (Balgaria) e grad ..." The statistical indicators will give a score 3 to the candidate "Balgaria" (Bulgaria), because it appears in three fragments which match Where-Is patterns.

4 Virtual database VIDA

The use of encyclopaedic resources in QA is a technique which was used successfully in some systems like START (`http://www.ai.mit.edu/projects/infolab/`) (Katz & Lin 2002) or MultiText (Clarke, Cormack & Kemkes 2003). Information Retrieval from such resources has usually higher precision than Information Retrieval from the entire web. Lin et al. (2003) points out that 47% of the TREC 2001 questions can be answered using dictionaries and encyclopaedias found in the web.

Encyclopaedic resources can be used offline or online. Online encyclopaedic resources are web dictionaries and the encyclopaedic Internet sites, such as biography.com or CIA factbook (Lin et.al. 2003). The problem with the offline

resources is that they do not change over time, therefore the information in them may soon become old. The online resources, on the other hand, can occasionally change their interface, disappear from the web, become slow to access and so on. The research in that field (Katz & Lin 2002) shows that the most successful approach is the one which switches between these two alternatives. Omnibase, used to backup the START QA system, creates local uniform indices of different online encyclopaedic resources.

We created the encyclopaedic database VIDA (VIrtual DAtabase) to serve as a knowledge source for Socrates. It is a collection of encyclopaedic resources accessible online or downloaded from the web. A unique index of all the resources was built which provides an interface both to offline and online resources.

While a lot of online encyclopaedias in different areas can be found in English language, it is more difficult to find such online information sources for Bulgarian. However, we discovered different web pages and sites which contain comprehensive information about historic events, actors, films, astronomy and other areas. We believe that as the web increases in size, the amount of such resources will increase.

Currently VIDA incorporates four encyclopaedic resources: a Calendar of the Historic Events, an Astronomical Dictionary, an Encyclopaedia of Actors and a Dictionary of Computer Abbreviations. The historic calendar and the computer abbreviations dictionary were downloaded from the web; the page with the main entries of the astronomical dictionary was also downloaded but only links to the pages which give more comprehensive details about the entries were stored in the database; the encyclopaedia of actors was not downloaded, instead we built an index with all the actors for which information is available in this site and stored in the database a link to the biographical page for every actor.

Different levels of indexing were applied to the different resources in VIDA. For three of the resources we indexed only the terms which are titles of the main entries and not the entries themselves. That is, we added to VIDA main index only the terms which appear as titles of entries. For example, for the astronomical dictionary we indexed the astronomical terms (sun, solar eclipse, Mars), (providing each term with a reference to its definition), but did not index the definitions. Therefore we use these three resources only for answers to definition questions. We processed and indexed all the words from the whole entries of the historical calendar, where every word is represented in the index with its basic form and its weight obtained via TF.IDF. This way, the historical calendar is used to reply to temporal questions.

4.1 *Answering temporal questions using VIDA*

VIDA incorporates a calendar of historic events which backups answer extraction for temporal questions.

While definition and Where-Is questions can be answered successfully searching on the web with patterns, temporal questions may have different syntactic structure and therefore it is not possible to build general patterns for the whole question class. We used two techniques for answering these questions. First, we perform vector search in VIDA — we search for answers to temporal questions in the Calendar of historic events. Next, we submit a query to Google API with the keywords from the question and rank the snippets according to the number of keywords which appear in them. We found that for the TREC 2001 temporal questions a simple vector search in the historic calendar worked much better than searching the answer with Google.

The Calendar of historic events represents a list of all the days in the year, each day has a list of important historic events which took place on that day. We pre-processed the entries of the calendar with the part-of-speech tagger of LINGUA (Tanev & Mitkov 2002) and built an index where every event is represented together with the date when it happened and a list of the base forms of the content words which describe the event, where every word is supplied with a weight calculated via TF.IDF. In this way a vector space model was implemented (Manning & Schütze 1999) to represent the events described in the Calendar. We also represent as a vector the question where for every content word weight is defined according to its part of speech and its capitalisation. This way if a question q is submitted to the system with keywords $\{q_i\}$, for every entry from the historic calendar with keywords $\{a_i\}$ the cosine between the vectors of the question and the calendar entry may be calculated using the following formula:

$$\frac{\sum_{i,j|q_i=a_j} weight(q_i).weight(a_j)}{\sqrt{\sum_i weight(q_i)^2}\sqrt{\sum_j weight(a_j)^2}}$$

where $weight(k)$ is the weight of the keyword k. Calendar entries which score above a certain threshold with this formula are suggested to the user as possible answers; the entries are ranked according to their score.

5 Combining patterns and encyclopaedic resources

Socrates searches answers to every question first in VIDA and then on the web. Results obtained from VIDA are ranked higher since the search in the encyclopaedic database has higher precision (this was demonstrated in our experiments on system performance evaluation). Similar approach for combining structured information sources and the web is adopted in Lin et al. (2003).

	Socrates	Baseline	Difference
Definition	0.447	0.338	0.109
Where-Is	0.417	0.191	0.226
Temporal	0.409	0.100	0.309
Total	0.433	0.261	0.172

Table 1: *MRR of Socrates, MRR of the baseline model and their difference*

6 Evaluation

We translated in Bulgarian 100 questions from TREC 2001: 61 definition questions, 22 temporal questions and 17 Where-Is questions.

We ran the system on this test set. We applied the metrics used by the TREC 2001 judges: We considered only the top five ranked answers. For every answer we calculated the Mean Reciprocal Rank (MRR). If the right answer was ranked first it is given score 1, if second — 0.5, if third — 0.333, if fourth — 0.25 and if fifth — 0.2. The total score from all the questions was then divided by the number of questions; in this way we obtained a MRR which was between 0 and 1.

Next we defined a baseline model: for every question we submitted to Google the query generated by the system for that question. We considered the top five snippets returned by Google and found the MRR for each Google response in the way described above.

Socrates answered correctly 52 questions obtaining a MRR of 0.43. The baseline model found answers to 38 questions and obtained a MRR of 0.26. Table 1 gives the MRR of the performance of Socrates, the baseline model and their difference by question type an in total. Socrates performs best on definitions questions. However, the baseline model has also high performance on this question class. Therefore, the difference between Socrates and the baseline model is about 0.11. Socrates has lowest performance on temporal questions, however it outperforms the baseline model with about 0.31 for this question type, which is the biggest difference between the system performance and the baseline model. The use of encyclopaedic resources countributes exclusively to the performance of the system on the temporal questions as all the correct responses to temporal questions were taken from the encyclopaedic database VIDA. In contrast, the answers to the defintion and Where-Is questions were extracted from the Google snippets using patterns and no answers were found in VIDA.

Table 2 shows examples of answers which the system has extracted from the web . The original questions and answers were in Bulgarian, here also their English translations are shown.

As it can be seen from the examples, Socrates returns sentences or fragments of sentences (when they are taken from the Google snippets) which con-

Question	Answer
Kakvo e epilepsiata?	Epilepsiata e hronichno zaboliavane na glavnia mozak
(What is epilepsy?)	(Epilepsy is a brain illness)
Koy e Duk Elingtan?	Duk Elingtan e nai-golemiat kompozitor v istoriata na jaza, koito...
(Who is Duke Ellington?)	(Duk Ellington is the greatest composer in the jazz history which...)
Kade e Efrat?	r.Efrat v severna mesopotamia
(Where is Euphrates?)	(Euphrates River in Northern Mesopotamia)
Kade e Golemiat Kanion?	niuyorkskia central park... Golemiat Kanion v Kolorado
(Where is the Great Canyon?)	(the New York Central Park... the Great Canyon in Colorado)

Table 2: *Examples of answers returned by* Socrates *(English translations in parenthesis)*

tain the answer rather than the exact answer itself. However, the presence of small pieces of redundant information (e.g., the last example in Table 2) does not affect seriously the clarity of answers representation.

7 Conclusions

We presented a prototype of a web QA system for the Bulgarian language. The system was able to answer more than half of the questions from the test set, its Mean Reciprocal Rank is 0.17 above the baseline model. The results are promising and show that pattern based Question Answering is efficient approach which should be studied further. The results demonstrate also that integration of encyclopaedic resources in a QA system can significantly increase its performance.

We intend to extend the range of the question types to which our system is able to answer. Automatic question type identifier and more precise question processor will be integrated in the system. Named entity recognition may improve the performance of Socrates. We intend to find synonyms and paraphrases of the most used Bulgarian verbs with corpus based techniques and to integrate this information into our system. We are also considering enlarging our encyclopaedic database VIDA with new resources. Finally, we intend to make our system available on the web as an alternative to the conventional search engines.

REFERENCES

Bucholz, Sabine & Walter Daelemans. 2001. "Complex Answers a Case Study Using a WWW Question-Answering System". *Question Answering* ed. by J. Tait, B. Boguraev, C. Jacquemin, R. Gaizauskas & L. Hirschman (= Special Issue of *Natural Language Engineering*) 7:4.301-323.

Clarke, Charles L., Gordon V. Cormack & Graeme Kemkes. 2003. "Statistical Selection of Exact Answers MultiText Experiments for TREC 2002". *Proceedings of the 11th Text Retrieval Conference* (*TREC 2002*), 823-831, Gaithesburg, Maryland.

Hirschman, Lynette & Robert Gaizauskas. 2001. "Natural Language Question Answering the View from Here". *Question Answering* ed. by J.Tait, B. Boguraev, C. Jacquemin, R. Gaizauskas & L. Hirschman (= Special Issue of *Natural Language Engineering*) 7:4.275-300.

Katz, Boris & Jimmy Lin. 2002. "START and Beyond". *Proceedings of the 6th World Multiconference on Systematics, Cybernetics and Informatics* (*SCI 2002*), 72-79. Orlando, Florida.

Lin, Jimmy, Aaron Fernandes, Boris Katz, Gregory Marton & Stefanie Tellex. 2003. "Extracting Answers from the Web Using Data Annotation and Knowledge Mining Techniques". *Proceedings of the 11th Text Retrieval Conference* (*TREC 2002*), 447-456. Gaithesburg, Maryland.

Manning, Christopher D. & Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, Mass.: MIT Press.

Ravichandran, Deepak. & Eduard Hovy. 2002. "Learning Surface Patterns for a Question Answering System". *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (*ACL'2002*), 41-47. Philadelphia, Pennsylvania, U.S.A.

Soubbotin, Martin & Sergei Soubbotin. 2002. "Patterns of Potential Answer Expressions as Clues to the Right Answers". *Proceedings of the 10th Text Retrieval Conference* (*TREC 2001*), 293-312. Gaithesburg, Maryland.

Tanev Hristo & Ruslan Mitkov. 2002. "Shallow Language Processing Architecture for Bulgarian". *Proceedings of the 19th International Conference on Computational Linguistics* (*COLING 2002*), vol.2, 995-1001. Taipei, Taiwan.

Voorhees, Ellen. 2001. "The TREC Question Answering Track". *Question Answering* ed. by J. Tait, B. Boguraev, C. Jacquemin, R. Gaizauskas & L. Hirschman (= Special Issue of *Natural Language Engineering*) 7:4.361-378.

Unsupervised Natural Language Disambiguation Using Non-Ambiguous Words

RADA MIHALCEA
University of North Texas

Abstract

This chapter describes an unsupervised approach for natural language disambiguation, applicable to ambiguity problems where classes of equivalence can be defined over the set of words in a lexicon. Lexical knowledge is induced from non-ambiguous words via classes of equivalence and enables the automatic generation of annotated corpora. The only requirements are a lexicon and a raw textual corpus. The method was tested on two natural language ambiguity tasks in several languages: part of speech tagging (English, Swedish, Chinese) and word sense disambiguation (English, Romanian). Classifiers trained on automatically constructed corpora were found to have a performance comparable with classifiers that learn from expensive manually annotated data.

1 Introduction

Ambiguity is inherent to human language. Successful solutions for automatic resolution of ambiguity in natural language often require large amounts of annotated data to achieve good levels of accuracy. While recent advances in Natural Language Processing (NLP) have brought significant improvements in the performance of NLP methods and algorithms, there has been relatively little progress on addressing the problem of obtaining annotated data required by some of the highest-performing algorithms. As a consequence, many of today's NLP applications experience severe data bottlenecks. According to recent studies (e.g., Banko & Brill 2001), the NLP research community should *"direct efforts toward increasing the size of annotated data collections"*, since large amounts of annotated data are likely to significantly impact the performance of current algorithms.

For instance, supervised part of speech tagging on English requires about 3 million words, each of them annotated with their corresponding part of speech, to achieve a performance in the range of 94-96%. State-of-the-art in English syntactic parsing is close to 88-89%, obtained by training parser models on a corpus of about 600,000 words, manually parsed within the Penn Treebank project, an annotation effort that required 2 man-years of work (Marcus 1993). Increased levels of problem complexity result in increasingly severe data bottlenecks. The data created so far for supervised English word sense disambiguation consist of tagged examples for about 200 ambiguous words. At a throughput of one tagged

example per minute (Edmonds 2000), with a requirement of about 500 tagged examples per word (Ng 1996), and with 20,000 ambiguous words in the common English vocabulary, this leads to about 160,000 hours of tagging – nothing less but 80 man-years of human annotation work. Information extraction, anaphora resolution and other tasks also strongly require large annotated corpora, which often are not available, or can be found only in limited quantities.

Moreover, problems related to lack of annotated data multiply by an order of magnitude when languages other than English are considered. The study of a new language (from about 7,200 different languages spoken worldwide) implies a similar amount of work in creating annotated corpora required by the supervised applications in the new language.

In this chapter, we describe a framework for unsupervised corpus annotation, applicable to ambiguity problems where classes of equivalence can be defined over the set of words in a lexicon. Part of speech tagging, word sense disambiguation, named entity disambiguation, are examples of such applications, where the same tag can be assigned to a set of words. In part of speech tagging, for instance, an *equivalence class* can be represented by the set of words that have the same functionality (e.g., noun). In word sense disambiguation, equivalence classes are formed by words with similar meaning (synonyms). The only requirements for this algorithm are a lexicon that defines the possible tags for each word, and a large raw corpus.

The underlying idea is based on the distinction between *ambiguous* and *non-ambiguous* words, and the knowledge that can be induced from the latter to the former via classes of equivalence. When building lexically annotated corpora, the main problem is represented by the words that, according to a given lexicon, have more than one possible tag. These words are *ambiguous* for the specific NLP problem. For instance, *"work"* is morphologically ambiguous, since it can be either a noun or a verb, depending on the context where it occurs. Similarly, *"plant"* carries on a semantic ambiguity, having both meanings of *"factory"* or *"living organism"*. Nonetheless, there are also words that carry only one possible tag, which are *non-ambiguous* for the given NLP problem. Since there is only one possible tag that can be assigned, the annotation of such *non-ambiguous* words can be accurately performed in an automatic fashion. Our method for unsupervised natural language disambiguation relies precisely on this latter type of words and on the equivalence classes that can be defined among words with similar tags.

Shortly, for an *ambiguous* word *W*, an attempt is made to identify one or more *non-ambiguous* words *W'* in the same class of equivalence, so that *W'* can be annotated in an automatic fashion. Next, lexical knowledge is induced from the *non-ambiguous* words *W'* to the *ambiguous words W* using classes of equivalence. The knowledge induction step is performed using a learning mechanism, where the automatically partially tagged corpus is used for training to annotate

new raw texts including instances of the ambiguous word *W*.

Related work in unsupervised or semi-supervised corpus annotation includes *active learning* (Dagan 1995), *co-training* (Blum & Mitchell 1998), *self-training* (Yarowsky 1995), *counter-training* (Yangarber 2003).

The chapter is organized as follows. We first present our unsupervised approach for building lexically annotated corpora and show how knowledge can be induced from *non-ambiguous* words via classes of equivalence. The method is evaluated on two natural language disambiguation tasks in several languages: part of speech tagging for English, Swedish and Chinese, and word sense disambiguation for English and Romanian.

2 Equivalence classes for building annotated corpora

The method introduced in this chapter relies on classes of equivalence defined among *ambiguous* and *non-ambiguous* words. The method assumes the availability of: (1) a lexicon that lists the possible tags a word might have, and (2) a large raw corpus. The algorithm consists of the following three main steps:

(1) Given a set $\mathcal{T}$ of possible tags, and a lexicon $\mathcal{L}$ with words W_i, i=1,$|\mathcal{L}|$, each word W_i admitting the tags T_j, j=1,$|W_i|$, determine equivalence classes $\mathcal{C}_{T_j}$, j=1,$|\mathcal{T}|$ containing all words that admit the tag T_j.

(2) Identify in the raw corpus all instances of words that belong to only one equivalence class. These are *non-ambiguous* words that represent the starting point for the annotation process. Each such *non-ambiguous* word is annotated with the corresponding tag from $\mathcal{T}$.

(3) The partially annotated corpus from step 2 is used to learn the knowledge required to annotate ambiguous words. Equivalence relations defined by the classes of equivalence $\mathcal{C}_{T_j}$ are used to determine *ambiguous* words W_i that are equivalent to the already annotated words. A label is assigned to each such ambiguous word by applying the following steps:

   (a) Detect all classes of equivalence $\mathcal{C}_{T_j}$ that include the word W_i.
   (b) In the corpus obtained at step 2, find all examples that are annotated with one of the tags T_j.
   (c) Use the examples from the previous step to form a training set, and use it to classify the current ambiguous instance W_i.

For illustration, consider the process of assigning a part of speech label to the word *"work"*, which may assume one of the labels NN (noun) or VB (verb). We identify in the corpus all instances of words already annotated with one of these two labels. These instances constitute training examples, annotated with one of the classes NN or VB. A classifier is then trained on these examples, and used to automatically assign a label to the current ambiguous word *"work"*. The

following sections detail on the features extracted from the context of a word to create training/test examples.

3 Evaluation

The method was evaluated on two natural language ambiguity problems. The first one is apart of speech tagging task, where a corpus annotated with part of speech tags is automatically constructed. The annotation accuracy of a classifier trained on automatically labeled data is compared against a baseline that assigns by default the most frequent tag, and against the accuracy of the same classifier trained on manually labeled data.

The second task is asemantic ambiguity problem, where the corpus construction method is used to generate a sense tagged corpus, which is then used to train a word sense disambiguation algorithm. The performance is again compared against the baseline, which assumes by default the most frequent sense, and against the performance achieved by the same disambiguation algorithm, trained on manually labeled data.

The precisions obtained during both evaluations are comparable with their alternatives relying on manually annotated data, and exceed by a large margin the simple baseline that assigns to each word the most frequent tag. Note that this baseline represents in fact a supervised classification algorithm, since it relies on the assumption that frequency estimates are available for tagged words.

Experiments were performed on several languages. The part of speech corpus annotation task was tested on English, Swedish and Chinese; the sense annotation task was tested on English and Romanian.

3.1 *Part of speech tagging*

The automatic annotation of a raw corpus with part of speech tags proceeds as follows. Given a lexicon that defines the possible morphological tags for each word, classes of equivalence are derived for each part of speech. Next, in the raw corpus, we identify and tag accordingly all the words that appear only in one equivalence class (i.e., *non-ambiguous* words). On average (as computed over several runs with various corpus sizes), about 75% of the words can be tagged at this stage. Using the equivalence classes, we identify ambiguous words in the corpus, which have one or more equivalent *non-ambiguous* words that were already tagged in the previous stage. Each occurrence of such *non-ambiguous* equivalents results in a training example. The training set derived in this way is used to classify the ambiguous instances.

For this task, a training example is formed using the following features:

(1) two words to the left and one word to the right of the target word, and their corresponding parts of speech (if available, or "?" otherwise);
(2) a flag indicating whether the current word starts with an uppercase letter;
(3) a flag indicating whether the current word contains any digits;
(4) the last three letters of the current word.

For learning, we use a memory based classifier — TiMBL (Daelemans 2001).

For each ambiguous word W_i defined in the lexicon, we determine all the classes of equivalence $\mathcal{C}_{T_j}$ to which it belongs, and identify in the training set all the examples that are labeled with one of the tags T_j. The classifier is then trained on these examples, and used to assign one of the labels T_j to the current instance of the ambiguous word W_i.

The unknown words (not defined in the lexicon) are labeled using a similar procedure, but this time assuming that the word may belong to any class of equivalence defined in the lexicon. Hence, the set of training examples is formed with all the examples derived from the partially annotated corpus.

The unsupervised part of speech annotation is evaluated in two ways. First, we compare the annotation accuracy with a simple baseline, that assigns by default the most frequent tag to each ambiguity class. Second, we compare the accuracy of the unsupervised method with the performance of the same tagging method, trained on manually labeled data. In all cases, we assume the availability of the same lexicon. Experiments and comparative evaluations are performed on English, Swedish and Chinese.

3.1.1 *Part of speech tagging for English*

For the experiments on English,we use the Penn Treebank Wall Street Journal part of speech tagged texts. Section 60, consisting of about 22,000 tokens, is set aside as a test corpus; the rest is used as a source of text data for training. The training corpus is cleaned of all part of speech tags, resulting in a raw corpus of about 3 million words. To identify classes of equivalence, we use a fairly large lexicon consisting of about 100,000 words with their corresponding parts of speech.

Several runs are performed, where the size of the lexically annotated corpus varies from as few as 10,000 tokens, up to 3 million tokens. In all runs, for both unsupervised or supervised algorithms, we use the same lexicon of about 100,000 words.

Table 1 lists results obtained for different training sizes. The table lists: the size of the training corpus, the part of speech tagging precision on the test data obtained with a classifier trained on (a) automatically labeled corpora, or (b) manually labeled corpora. For a 3 million words corpus, the classifier relying on manually annotated data outperforms the tagger trained on automatically constructed examples by 2.3%. There is practically no cost associated with the latter

Training corpus size	Evaluation on test set: Training corpus built automatically	Training corpus built manually
0 (baseline)	88.37%	
10,000	92.17%	94.04%
100,000	92.78%	94.84%
500,000	93.31%	95.76%
1,000,000	93.31%	96.54%
3,000,000	93.52%	95.88%

Table 1: *Corpus size and precision on test set using automatically or manually tagged training data* (*English*)

tagger, other than the requirement of obtaining a lexicon and a raw corpus, which eventually pays off for the slightly smaller performance.

3.1.2 *Part of speech tagging for Swedish*

For the Swedishpart of speech tagging experiment, we use text collections ranging from 10,000 words up to to 1 million words. We use the Stockholm Umea Corpus (SUC) corpus (`www.ling.su.se/staff/sofia/suc/suc.html`), and again a lexicon of about 100,000 words. The tagset is the one defined in SUC, and consists of 25 different tags.

As with the previous English-based experiments, the corpus is cleaned of part of speech tags, and run through the automatic labeling procedure. Table 2 lists the results obtained using corpora of various sizes. The accuracy continues to grow as the size of the training corpus increases, suggesting that larger corpora are expected to lead to higher precisions.

Training corpus size	Evaluation on test set: Training corpus build automatically	Training corpus build manually
0 (baseline)	83.07%	
10,000	87.28%	91.32%
100,000	88.43%	92.93%
500,000	89.20%	93.17%
1,000,000	90.02%	93.55%

Table 2: *Corpus size and precision on test set using automatically or manually tagged training data* (*Swedish*)

A similar experiment was run for Chinese.A lexicon of about 10,000 entries was derived from the Chinese Treebank. Using manually labeled data (an annotated corpus of about 100,000 tokens), an accuracy of 87.5% was measured on a data set of about 10,000 tokens. With the automatically labeled corpus, the performance on the same set was measured at 82.05%. When the raw training set is

augmented to about 2 million tokens, the precision of unsupervised annotation raises to 87.05%.

The conclusion drawn from these three experiments is that non-ambiguous words represent a useful source of knowledge for the task of part of speech tagging. The results are comparable with previously explored methods in unsupervised part of speech tagging: Cutting (1992) and Brill (1995) report a precision of 95-96% for part of speech tagging for English, using unsupervised annotation, under the assumption that all words in the test set are known. Under a similar assumption (i.e., all words in the test set are included in the lexicon), the performance of our unsupervised approach raises to 95.2%.

3.2 *Word sense disambiguation*

The annotation method was also evaluated on a word sense disambiguation problem. Here, the equivalence classes consist of words that are semantically related. Such semantic relations are often readily encoded in semantic networks, e.g., WordNet or EuroWordNet, and can be induced using bilingual dictionaries.

First, one or more *non-ambiguous* equivalents are identified for each possible meaning of the ambiguous word considered. For instance, the noun *"plant"*, with the two meanings of *"living organism"* and *"manufacturing plant"*, has the monosemous equivalents *"flora"* and *"industrial plant"*.

Next, the monosemous equivalents are used to extract several examples from a raw textual corpus, which constitute training examples for the semantic annotation task. The feature set used for this task consists of a surrounding window of two words to the left and right of the target word, the verbs before and after the target word, the nouns before and after the target word, and sense specific keywords. Similar with the experiments on part of speech tagging, we use the TiMBL memory based learner.

The performance obtained with the automatically tagged corpus is evaluated against: (1) a simple baseline, which assigns by default the most frequent sense (as determined from the training corpus); and (2) a supervised method that learns from manually annotated corpora (the performance of the supervised method is estimated through ten-fold cross validations)

For the English task, *monosemous* equivalents for six ambiguous words were determined based on WordNet synsets. The raw corpus consists of the British National Corpus, with about 100 million words, containing news article, scientific reports, novels, and spoken transcripts. Each monosemous equivalent is used to extract several examples (consisting of 4 sentences surrounding the focus word), up to a maximum of 100 examples per word sense.

Table 3 lists the six ambiguous words, the size of the training corpus automatically generated, the precision obtained on the test set using: (1) a simple heuristic that assigns the most frequent sense to all instances; (2) the classifier

Word	Train. corpus	Test size	Most freq. sense	Disambig. precision Training corpus auto.	Training corpus manual
bass	107	107	90.65%	92.52%	90.65%
crane	200	95	74.73%	71.57%	81.05%
motion	200	200	70.14%	75.62%	88.05%
palm	200	200	71.14%	80.59%	87.06%
plant	200	188	54.36%	69.14%	76.59%
tank	100	200	62.69%	63.69%	76.61%
AVERAGE	184	171	70.61%	76.60%	83.35%

Table 3: *Corpus size, disambiguation precision using most frequent sense, and using automatically or manually sense tagged data (English)*

trained on (2a) automatically generated corpora, or (2b) manually labeled data.[1]

The disambiguation accuracy clearly exceeds the baseline, even for such small amounts of annotated corpora. While previous results reported in the literature for these words are sometimes larger (e.g., Yarowsky (1995)), note that the size of our corpus is limited to at most 100 examples per word sense, to allow for a one-to-one comparison with supervised methods (as opposed to thousands of annotated examples). Moreover, to avoid the bias introduced by the "one sense per discourse" paradigm, the examples that were manually annotated were selected exclusively from individual BNC texts.

Similar disambiguation experiments were also performed on Romanian.Since a Romanian WordNet is not yet available, *monosemous* equivalents for five ambiguous words were hand-picked by a native speaker using a paper-based dictionary. The raw corpus consists of a collection of Romanian newspapers collected on the Web over a three years period (1999-2002).

The monosemous equivalents are used to extract several examples, again with a surrounding window of 4 sentences. An interesting problem that occurred in this task is the gender, which may influence the classification decision. To avoid possible miss-classifications due to gender mismatch, the native speaker was instructed to pick the monosemous equivalents such that they all have the same gender (which is not necessarily the gender of their equivalent *ambiguous* word). Next, the *non-ambiguous* words are replaced with their *ambiguous* equivalent, and consequently we obtain a corpus annotated with sense tags.

Table 4 lists the five ambiguous words, their monosemous equivalents, the size of the training corpus automatically generated, and the precision obtained on the test set using the simple most frequent sense heuristic and the instance based classifier. Again, the classifier trained on the automatically labeled data exceeds by a large margin the simple heuristic that assigns the most frequent sense by

[1] The manually annotated corpus for these words is available from `http://www.cs.unt.edu/~rada/downloads.html`

default. Since the size of the test set created for these words is fairly small (50 examples or less for each word), the performance of a supervised method could not be estimated.

Word	English translations	Training size	Most freq. sense	Disambig. precision
volum	book/quantity	200	52.85%	87.05%
galerie	museum/tunnel	200	66.00%	80.00%
canal	channel/tube	200	69.62%	95.47%
slujba	job/service	67	58.8%	83.3%
vas	container/ship	164	60.9%	91.3%
	AVERAGE	166	61.63%	87.42%

Table 4: *Corpus size, disambiguation precision using most frequent sense, and using automatically sense tagged data (Romanian)*

For an average training size of 164 examples per word, the annotation accuracy for this data set of five words was evaluated at 87.42%, compared to the most frequent sense baseline of 61.63%.

4 Conclusions

This chapter introduced a framework for unsupervised natural language disambiguation, applicable to ambiguity problems where classes of equivalence can be defined over the set of words in a lexicon. Lexical knowledge is induced from non-ambiguous words via classes of equivalence, and enables the automatic generation of annotated corpora. The only requirements are a dictionary and a raw textual corpus. The method was tested on two natural language ambiguity tasks, on several languages. In part of speech tagging, classifiers trained on automatically constructed training corpora performed at accuracies in the range of 88-94%, depending on training size, comparable with the performance of the same tagger when trained on manually labeled data. Similarly, in word sense disambiguation experiments, the algorithm succeeds in creating semantically annotated corpora, which enable good disambiguation accuracies. In future work, we plan to investigate the application of this algorithm to very, very large corpora, and evaluate the impact on disambiguation performance.

Acknowledgements. This work was partially supported by a National Science Foundation grant IIS-0336793.

REFERENCES

Banko, Michele & Eric Brill. 2001. "Scaling to Very Very Large Corpora for Natural Language Disambiguation". *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (*ACL'01*), 26-33. Toulouse, France.

Blum, Avrim & Tom Mitchell. 1998. "Combining Labeled and Unlabeled Data with Co-training". *Proceedings of the Workshop on Computational Learning Theory* (*COLT 1998*), 92-100. Madison, Wisc.

Brill, Eric. 1995. "Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging". *Proceedings of the ACL Third Workshop on Very Large Corpora*, 1-13. Somerset, N.J.

Cutting, Doug, Julian Kupiec, Jan Pedersen & Penelope Sibun. 1992. "A Practical Part-of-Speech Tagger". *Proceedings of the Third Conference on Applied Natural Language Processing* (*ANLP-92*), 133-140. Trento, Italy.

Daelemans, Walter, Jakub Zavrel, Ko van der Sloot & Antal van den Bosch. 2001. "TiMBL: Tilburg Memory Based Learner, version 4.0, reference guide". Technical report. Antwerp, Belgium: Univ. of Antwerp.

Dagan, Ido & Sean P. Engelson. 1995. "Committee-based Sampling for Training Probabilistic Classifiers". *Proceedings of the International Conference on Machine Learning* (*ICML-1995*), 150-157. Tahoe City, Calif.

Edmonds, Phil. 2000. "Designing a Task for Senseval-2". Technical Note, *Senseval-2 Workshop*, Toulouse, France.

Marcus, Mitch, Beatrice Santorini & Mary Ann Marcinkiewicz. 1993. "Building a Large Annotated Corpus of English: The Penn Treebank". *Computational Linguistics* 19:2.313-330.

Ng, Hwee Tou & Hian Beng Lee. 1996. "Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-based Approach". *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics* (*ACL'96*), 40-57. Santa Cruz, Calif.

Yangarber, Roman. 2003. "Counter-Training in Discovery of Semantic Patterns". *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics* (*ACL'03*), 343-350. Sapporo, Japan.

Yarowsky, David. 1995. "Unsupervised Word Sense Disambiguation Rivaling Supervised Methods". *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics* (*ACL'95*), 189-196. Cambridge, Mass.

List of Contributors

Eneko Agirre *eneko@si.ehu.es*

Galia Angelova *galia@lml.bas.bg*

Amit Bagga *bagga@avaya.com*

Roberto Basili*basili@info.uniroma2.it*

Jeffrey Bigham *jbigham@u.washington.edu*

Branimir K. Boguraev *bran@us.ibm.com*

Yuri Bonev *y.bonev@team-vision.bg*

Svetla Boytcheva *svetla@fmi.uni-sofia.bg*

António Branco *ahb@di.fc.ul.pt*

Xavier Carreras *carreras@lsi.upc.es*

Roberta Catizone *r.catizone@dcs.shef.ac.uk*

Timothy Chklovski *timc@isi.edu*

Henning Christiansen *henning@ruc.dk*

Hamish Cunningham *hamish@dcs.shef.ac.uk*

Ido Dagan *dagan@cs.biu.ac.il*

Sebastian van Delden *sdelden@cs.ucf.edu*

Richard J. Evans *R.J.Evans@wlv.ac.uk*

Elena Filatova *filatova@cs.columbia.edu*

Chris Fox *foxcj@essex.ac.uk*

Jesús Giménez *jgimenez@lsi.upc.es*

Oren Glickman *glikmao@cs.biu.ac.il*

Fernando Gomez *gomez@cs.ucf.edu*

Evelyn Gius *evelyn.gius@imail.de*

Walther v. Hahn
vhahn@informatik.uni-hamburg.de

Vasileios Hatzivassiloglou
vh@cs.columbia.edu

Mary Hearne *mhearne@computing.dcu.ie*

Mark Hepple *m.hepple@dcs.shef.ac.uk*

Graeme Hirst *gh@cs.toronto.edu*

Anette Hulth *hulth@dsv.su.se*

Camelia Ignat *Camelia.Ignat@jrc.it*

Diana Zaiu Inkpen *diana@site.uottawa.ca*

Mario Jarmasz *Mario.Jarmasz@cnrc-nrc.gc.ca*

Sandra Kübler *kuebler@sfs.uni-tuebingen.de*

Oier Lopez de Lacalle *jibloleo@si.ehu.es*

Shalom Lappin *lappin@dcs.kcl.ac.uk*

Maria Liakata *maria.liakata@clg.ox.ac.uk*

Michael L. Littman *mlittman@cs.rutgers.edu*

Susannah J. Lydon *susannah@cs.man.ac.uk*

Inderjeet Mani *im5@georgetown.edu*

Hanady Mansour *hanady@ccl.umist.ac.uk*

Lluís Màrquez *lluism@lsi.upc.es*

Diana Maynard *diana@dcs.shef.ac.uk*

Rada Mihalcea *rada@cs.unt.edu*

Tristan Miller *tristan.miller@dfki.de*

Preslav Nakov *nakov@cs.berkeley.edu*

Ani Nenkova *ani@cs.columbia.edu*

Luka Nerima *Luka.Nerima@lettres.unige.ch*

Leif Arda Nielsen *nielsen@dcs.kcl.ac.uk*

Maria Teresa Pazienza *pazienza@info.uniroma2.it*

Leonid Peshkin *pesha@eecs.harvard.edu*

Bruno Pouliquen *Bruno.Pouliquen@jrc.it*

Stephen G. Pulman *Stephen.Pulman@clg.ox.ac.uk*

Allan Ramsay *Allan.Ramsay@manchester.ac.uk*

Virginia Savova *savova@jhu.edu*

Violeta Seretan *Violeta.Seretan@lettres.unige.ch*

Andrea Setzer *a.setzer@dcs.shef.ac.uk*

Victor Shnayder *shnayder@eecs.harvard.edu*

João Silva *jsilva@di.fc.ul.pt*

Khalil Sima'an *simaan@science.uva.nl*

Kiril Iv. Simov *kivs@bultreebank.org*

Ralf Steinberger *Ralf.Steinberger@jrc.it*

Albena Strupchanska *albena@lml.bas.bg*

Stan Szpakowicz *szpak@site.uottawa.ca*

Valentin Tablan *valyt@dcs.shef.ac.uk*

Hristo T. Tanev *tanev@itc.it*

Peter D. Turney *peter.turney@nrc.ca*

Elena Valchanova *elenav@lml.bas.bg*

Nick Webb *n.webb@dcs.shef.ac.uk*

Eric Wehrli *Eric.Wehrli@lettres.unige.ch*

Yorick Wilks *y.wilks@dcs.shef.ac.uk*

Mary McGee Wood *mary@cs.man.ac.uk*

Milena Yankova *myankova@lml.bas.bg*

Fabio Massimo Zanzotto *zanzotto@info.uniroma2.it*

Index of Subjects and Terms

CURRENT ISSUES IN LINGUISTIC THEORY

E. F. K. Koerner, Editor
Zentrum für Allgemeine Sprachwissenschaft, Typologie
und Universalienforschung, Berlin
efk.koerner@rz.hu-berlin.de

Current Issues in Linguistic Theory (CILT) is a theory-oriented series which welcomes contributions from scholars who have significant proposals to make towards the advancement of our understanding of language, its structure, functioning and development. CILT has been established in order to provide a forum for the presentation and discussion of linguistic opinions of scholars who do not necessarily accept the prevailing mode of thought in linguistic science. It offers an outlet for meaningful contributions to the current linguistic debate, and furnishes the diversity of opinion which a healthy discipline must have. A complete list of titles in this series can be found on the publishers website, **www.benjamins.com**

222 **HERSCHENSOHN, Julia, Enrique MALLÉN and Karen ZAGONA (eds.):** Features and Interfaces in Romance. Essays in honor of Heles Contreras. 2001. xiv, 302 pp.

223 **FANEGO, Teresa, Javier PÉREZ-GUERRA and María José LÓPEZ-COUSO (eds.):** English Historical Syntax and Morphology. Selected papers from 11 ICEHL, Santiago de Compostela, 7–11 September 2000. Volume 1. 2002. x, 306 pp.

224 **FANEGO, Teresa, Belén MÉNDEZ-NAYA and Elena SEOANE (eds.):** Sounds, Words, Texts and Change. Selected papers from 11 ICEHL, Santiago de Compostela, 7–11 September 2000. Volume 2. 2002. x, 310 pp.

225 **SHAHIN, Kimary N.:** Postvelar Harmony. 2003. viii, 344 pp.

226 **LEVIN, Saul:** Semitic and Indo-European. Volume II: Comparative morphology, syntax and phonetics. 2002. xviii, 592 pp.

227 **FAVA, Elisabetta (ed.):** Clinical Linguistics. Theory and applications in speech pathology and therapy. 2002. xxiv, 353 pp.

228 **NEVIN, Bruce E. (ed.):** The Legacy of Zellig Harris. Language and information into the 21st century. Volume 1: Philosophy of science, syntax and semantics. 2002. xxxvi, 323 pp.

229 **NEVIN, Bruce E. and Stephen B. JOHNSON (eds.):** The Legacy of Zellig Harris. Language and information into the 21st century. Volume 2: Mathematics and computability of language. 2002. xx, 312 pp.

230 **PARKINSON, Dilworth B. and Elabbas BENMAMOUN (eds.):** Perspectives on Arabic Linguistics. Papers from the Annual Symposium on Arabic Linguistics. Volume XIII-XIV: Stanford, 1999 and Berkeley, California 2000. 2002. xiv, 250 pp.

231 **CRAVENS, Thomas D.:** Comparative Historical Dialectology. Italo-Romance clues to Ibero-Romance sound change. 2002. xii, 163 pp.

232 **BEYSSADE, Claire, Reineke BOK-BENNEMA, Frank DRIJKONINGEN and Paola MONACHESI (eds.):** Romance Languages and Linguistic Theory 2000. Selected papers from 'Going Romance' 2000, Utrecht, 30 November–2 December. 2002. viii, 354 pp.

233 **WEIJER, Jeroen van de, Vincent J. van HEUVEN and Harry van der HULST (eds.):** The Phonological Spectrum. Volume I: Segmental structure. 2003. x, 308 pp.

234 **WEIJER, Jeroen van de, Vincent J. van HEUVEN and Harry van der HULST (eds.):** The Phonological Spectrum. Volume II: Suprasegmental structure. 2003. x, 264 pp.

235 **LINN, Andrew R. and Nicola McLELLAND (eds.):** Standardization. Studies from the Germanic languages. 2002. xii, 258 pp.

236 **SIMON-VANDENBERGEN, Anne-Marie, Miriam TAVERNIERS and Louise J. RAVELLI (eds.):** Grammatical Metaphor. Views from systemic functional linguistics. 2003. vi, 453 pp.

237 **BLAKE, Barry J. and Kate BURRIDGE (eds.):** Historical Linguistics 2001. Selected papers from the 15th International Conference on Historical Linguistics, Melbourne, 13–17 August 2001. Editorial Assistant: Jo Taylor. 2003. x, 444 pp.

238 **NÚÑEZ-CEDEÑO, Rafael, Luis LÓPEZ and Richard CAMERON (eds.):** A Romance Perspective on Language Knowledge and Use. Selected papers from the 31st Linguistic Symposium on Romance Languages (LSRL), Chicago, 19–22 April 2001. 2003. xvi, 386 pp.

239 **ANDERSEN, Henning (ed.):** Language Contacts in Prehistory. Studies in Stratigraphy. Papers from the Workshop on Linguistic Stratigraphy and Prehistory at the Fifteenth International Conference on Historical Linguistics, Melbourne, 17 August 2001. 2003. viii, 292 pp.

240 **JANSE, Mark and Sijmen TOL (eds.):** Language Death and Language Maintenance. Theoretical, practical and descriptive approaches. With the assistance of Vincent Hendriks. 2003. xviii, 244 pp.

241 **LECARME, Jacqueline (ed.):** Research in Afroasiatic Grammar II. Selected papers from the Fifth Conference on Afroasiatic Languages, Paris, 2000. 2003. viii, 550 pp.

242 **SEUREN, Pieter A.M. and Gerard KEMPEN (eds.):** Verb Constructions in German and Dutch. 2003. vi, 316 pp.

243 **CUYCKENS, Hubert, Thomas BERG, René DIRVEN and Klaus-Uwe PANTHER (eds.):** Motivation in Language. Studies in honor of Günter Radden. 2003. xxvi, 403 pp.

244 **PÉREZ-LEROUX, Ana Teresa and Yves ROBERGE (eds.):** Romance Linguistics. Theory and Acquisition. Selected papers from the 32nd Linguistic Symposium on Romance Languages (LSRL), Toronto, April 2002. 2003. viii, 388 pp.

245 **QUER, Josep, Jan SCHROTEN, Mauro SCORRETTI, Petra SLEEMAN and Els VERHEUGD (eds.):** Romance Languages and Linguistic Theory 2001. Selected papers from 'Going Romance', Amsterdam, 6–8 December 2001. 2003. viii, 355 pp.

246 **HOLISKY, Dee Ann and Kevin TUITE (eds.):** Current Trends in Caucasian, East European and Inner Asian Linguistics. Papers in honor of Howard I. Aronson. 2003. xxviii, 426 pp.

247 **PARKINSON, Dilworth B. and Samira FARWANEH (eds.):** Perspectives on Arabic Linguistics XV. Papers from the Fifteenth Annual Symposium on Arabic Linguistics, Salt Lake City 2001. 2003. x, 214 pp.

248 **WEIGAND, Edda (ed.):** Emotion in Dialogic Interaction. Advances in the complex. 2004. xii, 284 pp.

249 **BOWERN, Claire and Harold KOCH (eds.):** Australian Languages. Classification and the comparative method. 2004. xii, 377 pp. (incl. CD-Rom).

250 **JENSEN, John T.:** Principles of Generative Phonology. An introduction. 2004. xii, 324 pp.

251 **KAY, Christian J., Simon HOROBIN and Jeremy J. SMITH (eds.):** New Perspectives on English Historical Linguistics. Selected papers from 12 ICEHL, Glasgow, 21–26 August 2002. Volume I: Syntax and Morphology. 2004. x, 264 pp.

252 **KAY, Christian J., Carole HOUGH and Irené WOTHERSPOON (eds.):** New Perspectives on English Historical Linguistics. Selected papers from 12 ICEHL, Glasgow, 21–26 August 2002. Volume II: Lexis and Transmission. 2004. xii, 273 pp.

253 **CAFFAREL, Alice, J.R. MARTIN and Christian M.I.M. MATTHIESSEN (eds.):** Language Typology. A functional perspective. 2004. xiv, 702 pp.

254 **BALDI, Philip and Pietro U. DINI (eds.):** Studies in Baltic and Indo-European Linguistics. In honor of William R. Schmalstieg. 2004. xlvi, 302 pp.

255 **MEULEN, Alice ter and Werner ABRAHAM (eds.):** The Composition of Meaning. From lexeme to discourse. 2004. vi, 232 pp.

256 **BOK-BENNEMA, Reineke, Bart HOLLEBRANDSE, Brigitte KAMPERS-MANHE and Petra SLEEMAN (eds.):** Romance Languages and Linguistic Theory 2002. Selected papers from 'Going Romance', Groningen, 28–30 November 2002. 2004. viii, 273 pp.

257 **FORTESCUE, Michael, Eva Skafte JENSEN, Jens Erik MOGENSEN and Lene SCHØSLER (eds.):** Historical Linguistics 2003. Selected papers from the 16th International Conference on Historical Linguistics, Copenhagen, 11–15 August 2003. 2005. x, 312 pp.

258 **AUGER, Julie, J. Clancy CLEMENTS and Barbara VANCE (eds.):** Contemporary Approaches to Romance Linguistics. Selected Papers from the 33rd Linguistic Symposium on Romance Languages (LSRL), Bloomington, Indiana, April 2003. With the assistance of Rachel T. Anderson. 2004. viii, 404 pp.

259 **CARR, Philip, Jacques DURAND and Colin J. EWEN (eds.):** Headhood, Elements, Specification and Contrastivity. Phonological papers in honour of John Anderson. 2005. 430 pp.

260 **NICOLOV, Nicolas, Kalina BONTCHEVA, Galia ANGELOVA and Ruslan MITKOV (eds.):** Recent Advances in Natural Language Processing III. Selected papers from RANLP 2003. 2004. xii, 402 pp.

261 **KAY, Christian J. and Jeremy J. SMITH (eds.):** Categorization in the History of English. 2004. viii, 268 pp.

262 **VAJDA, Edward J. (ed.):** Languages and Prehistory of Central Siberia. 2004. x, 275 pp.

263 **BRANCO, António, Tony McENERY and Ruslan MITKOV (eds.):** Anaphora Processing. Linguistic, cognitive and computational modelling. 2005. x, 449 pp.

264 **DRESSLER, Wolfgang U., Dieter KASTOVSKY, Oskar E. PFEIFFER and Franz RAINER (eds.):** Morphology and its demarcations. Selected papers from the 11th Morphology meeting, Vienna, February 2004. ca. 345 pp. *Expected Summer 2005*

265 **CORNIPS, Leonie and Karen P. CORRIGAN (eds.):** Syntax and Variation. Reconciling the Biological and the Social. ca. 265 pp. *Expected Summer 2005*