Lecture Notes in Artificial Intelligence 2703
Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Osmar R. Zaïane Jaideep Srivastava
Myra Spiliopoulou Brij Masand (Eds.)

WEBKDD 2002 – Mining Web Data for Discovering Usage Patterns and Profiles

4th International Workshop
Edmonton, Canada, July 23, 2002
Revised Papers

Springer

Volume Editors

Osmar R. Zaïane
University of Alberta, Department of Computing Science
Edmonton, Alberta, T6G 2E8 Canada
E-mail: zaiane@cs.ualberta.ca

Jaideep Srivastava
University of Minnesota, Computer Science and Engineering
Minneapolis, MN 55455, USA
E-mail: srivasta@cs.umn.edu

Myra Spiliopoulou
Otto-von-Guericke University of Magdeburg, Faculty of Computer Science
Institute of Technical and Business Information Systems
P.O. Box 4120, 39016 Magdeburg, Germany
E-mail: myra@iti.cs.uni-magdeburg.de

Brij Masand
Data Miners Inc.
77 North Washington Street, 9th Floor, Boston, MA 02114, USA
E-mail: brij@data-miners.com

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): I.2, H.2.8, H.3-4, K.4, C.2

ISSN 0302-9743
ISBN 3-540-20304-4 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg NewYork
a member of BertelsmannSpringer Science+Business Media GmbH

www.springeronline.com

Typesetting: Camera-ready by author, data conversion by PTP-Berlin GmbH
Printed on acid-free paper SPIN: 10928684 06/3142 5 4 3 2 1 0

Preface

1 Workshop Theme

Data mining as a discipline aims to relate the analysis of large amounts of user data to shed light on key business questions. Web usage mining in particular, a relatively young discipline, investigates methodologies and techniques that address the unique challenges of discovering insights from Web usage data, aiming to evaluate Web usability, understand the interests and expectations of users and assess the effectiveness of content delivery. The maturing and expanding Web presents a key driving force in the rapid growth of electronic commerce and a new channel for content providers. Customized offers and content, made possible by discovered knowledge about the customer, are fundamental for the establishment of viable e-commerce solutions and sustained and effective content delivery in noncommercial domains. Rich Web logs provide companies with data about their online visitors and prospective customers, allowing microsegmentation and personalized interactions.

While Web mining as a domain is several years old, the challenges that characterize data analysis in this area continue to be formidable. Though preprocessing data routinely takes up a major part of the effort in data mining, Web usage data presents further challenges based on the difficulties of assigning data streams to unique users and tracking them over time. New innovations are required to reliably reconstruct sessions, to ascertain similarity and differences between sessions, and to be able to segment online users into relevant groups. While Web usage data is large in volume, recommender system approaches still suffer from the challenges of compiling enough data to correlate user preferences to products. Intelligent use of domain abstractions that can help characterize how to group Web pages and incorporating knowledge of Web site structure into Web data analysis are new areas that can take Web usage mining to the next level. This workshop addresses advances along these lines as demonstrated by the papers included in this volume.

WEBKDD 2002 was the fourth in the WEBKDD series of workshops, with special emphasis on mining Web data for discovering usage patterns and profiles. WEBKDD 1999 focused on the aspects of Web mining related to user profiling, WEBKDD 2000 focused on Web Mining for E-Commerce, and WEBKDD 2001 focused on mining Web log data across all customer touchpoints.

The KDD community responded very enthusiastically to the WEBKDD 2002 workshop. More than 50 people attended the workshop, which brought together practitioners, tool vendors and researchers interested in Web mining. The paper presentations were divided into three sessions, titled "Categorization of Users and Usage," "Prediction and recommendation," and "Evaluation of algorithms." A total of 23 papers were submitted to WEBKDD 2002, of which 10 were selected for presentation at the workshop – a 44% acceptance rate. The authors of the

papers presented at WEBKDD 2002 were invited to submit extended versions of their papers for this special issue. A second round of review was carried out for each paper, and the revised and enhanced versions of all 10 papers are included in this book. In the next section we summarize each paper.

The final versions of the workshop papers can be found in the online repository of published papers: http://www.acm.org/sigkdd/proceedings/webkdd02/.

2 Papers

The first presentation, by Chi, Rosien and Heer investigated whether major groupings of user traffic on a Web site can be discovered in an automated fashion. In their paper "LumberJack: Intelligent Discovery and Analysis of Web User Traffic Composition," they describe how using multiple features from the user session data, automated clustering analysis can achieve high classification accuracy.

In their paper "Mining eBay: Bidding Strategies and Shill Detection," Shah, Joshi, Sureka, and Wurman investigate bidding strategies in online auction markets. Using online auction data from eBay for video game console auctions, they propose new attributes of bidding engagements and rules for classifying strategies. Through the analysis of auctions where multiple sessions for individuals are tracked over time, analysis of the bidding sessions led them to identify some known and new bidding behaviors, including clustering of strategies. Among the bidding strategies they identify is shilling behaviour (where there is a fake orchestrated sequence of bids to drive up the price).

In their presentation on "Automatic Categorization of Web Pages and User Clustering by Using Mixtures of Hidden Markov Models," Ypma and Heskes describe an EM algorithm for training the mixture of HMMs and also show how to use prior knowledge to help the learning process. They test their algorithm on both artificial data where they demonstrate that the correct patterns are being learnt, as well as on real data from a commercial Web site.

In their presentation on "Web Usage Mining by Means of Multidimensional Sequence Alignment Methods," Hay, Wets and Vanhoof explore the analysis of Web usage sequences by using a multidimensional sequence alignment method. They illustrate how navigation patterns are discovered by aligning sequences by using not only page sequence but also the time visited per page. They demonstrate empirical test results of the new algorithm, MDSAM, which show discovered user profiles.

In their paper "A Customizable Behavior Model for Temporal Prediction of Web User Sequences," Frias-Martinez and Karamcheti model the behavior of users for the prediction of sequences of Web user access. They propose a new model that allows customizing of the sequential nature of the preceding (known) patterns as well as the predicted patterns. Their algorithm also calculates a measure of the gap between the antecedent and the consequent sequences.

In his paper "Coping with Sparsity in a Recommender System," André Bergholz addresses the important problem of sparseness in the data used for

recommender systems. Typically there are too few ratings, resulting in limited correlations between users. Bergholz explores two innovative approaches for resolving this problem. The first one uses transitive correlations between existing users, which adds new ratings without much computational expense. The second approach uses a ratings agent that actually assigns ratings in accordance with some predefined preferences, resulting in increased coverage but some impact on system performance.

In their presentation "On the Use of Constrained Associations for Web Log Mining," Yang and Parthasarthy investigated the problem of scalability when mining for association rules for predicting patterns of Web usage access. They describe a new approach for mining such rules which introduces constraints in the way the rules are discovered. They explore the hypothesis that considering recent pages is more important in predicting consequent pages. This ordering and temporal constraint results in simpler and fewer rules, thus enabling faster online deployment.

In their paper "Mining WWW Access Sequence by Matrix Clustering," Oyanagi, Kubota and Nakase present a novel approach to sequence mining using matrix clustering. Their method, which results in generalized sequence patterns, decomposes a sequence into a set of sequence elements, each of which corresponds to an ordered pair of items. Then matrix clustering is applied to extract a cluster of similar sequences. The resulting sequence elements are then combined into generalized sequences. The authors demonstrate the discovery of long sequences in actual Web logs.

In their presentation on "Comparing Two Recommender Algorithms with the Help of Recommendations by Peers," Geyer-Schulz and Hahsler presented a framework for evaluating the effectiveness of recommender systems. They investigate whether the recommendations produced by algorithms such as those using frequent itemsets are useful for the social process of making further recommendations to others. They use two algorithms, one using association rules and another using repeat-buying theory from marketing research to compare how well the recommendations produced by brokers in an information market firm compare with those that are produced by the algorithm. They find that the recommendations compare quite favorably, with a high degree of accuracy and precision, and that both algorithms perform similarly when tuned appropriately on real data.

In their paper on "The Impact of Site Structure and Web Environment on Session Reconstruction in Web Usage Analysis," Berendt, Mobasher, Nakagawa and Spiliopoulou investigate the reliability of session reconstruction, and continue their work on characterizing the issues in Web preprocessing. They investigate factors such as the presence and absence of cookies, server-side session information, effect of site structure, etc., to evaluate errors introduced in session reconstruction and its subsequent impact on predictions for Web personalization. They specifically analyze data from frame-based and frame-free versions of a site with respect to the quality of session reconstruction using different heuristics,

and suggest that Web site characteristics can be a guide to the use of specific heuristics.

3 Conclusion

In its fourth year as a workshop, WEBKDD 2002 continued to show the sustained interest in this area and turned out to be a very successful workshop. About 50 people attended it, roughly divided evenly between industry and academia. It was an interactive workshop with a lively exchange between presenters and attendees. This year, apart from the continuing themes of new approaches to analyze and cluster sessions, there were more papers on recommender systems. We hope to be more inclusive in future years to expand the range of research topics included in the workshop.

4 Acknowledgements

We would like to acknowledge the Program Committee members of WEBKDD 2002 who invested their time in carefully reviewing papers for this volume: Jonathan Becher (Accrue/Neovista Software, Inc.), Bettina Berendt (HU Berlin, Germany), Alex Buechner (Lumio Ltd. UK), Ed Chi (Xerox Parc, USA), Robert Cooley (KXEN, USA), Wolfgang Gaul (University Karlsruhe, Germany), Oliver Guenther (HU Berlin, Germany), Ronny Kohavi (Blue Martini Software, USA), Vipin Kumar (AHPCRC-University of Minnesota, USA), Ee-Peng Lim (Chinese University of Hong Kong, China), Sanjay Kumar Madria (University of Missouri-Rolla), Yannis Manolopoulos (Aristotle Univ., Greece), Bamshad Mobasher (De Paul Univ., USA), Jian Pei (SUNY Buffalo, USA), Alex Tuzhilin (NYU/Stern School of Business, USA), Terry Woodfield (SAS Institute, Inc.), and Mohammed Zaki (Rensselaer Polytechnic Institute, USA).

We would also like to thank others who contributed to WEBKDD 2002, including the original PC members who reviewed the first set of workshop papers. We are grateful to the KDD 2002 organizing committee, especially Renée Miller, University of Toronto (Workshops Chair), Jörg Sander, University of Alberta, Canada (Registration Chair) and Mario Nascimento, University of Alberta, Canada (Local Arrangements Chair), for their help in bringing the WEBKDD community together. Finally we would like to thank the many participants who brought their ideas, research and enthusiasm to the workshop and proposed many new directions for WEBKDD research.

June 2003

Osmar R. Zaïane
Jaideep Srivastava
Myra Spiliopoulou
Brij Masand

Table of Contents

LumberJack: Intelligent Discovery and Analysis of Web User Traffic Composition 1
Ed H. Chi, Adam Rosien, Jeffrey Heer

Mining eBay: Bidding Strategies and Shill Detection 17
Harshit S. Shah, Neeraj R. Joshi, Ashish Sureka, Peter R. Wurman

Automatic Categorization of Web Pages and User Clustering with Mixtures of Hidden Markov Models 35
Alexander Ypma, Tom Heskes

Web Usage Mining by Means of Multidimensional Sequence Alignment Methods 50
Birgit Hay, Geert Wets, Koen Vanhoof

A Customizable Behavior Model for Temporal Prediction of Web User Sequences 66
Enrique Frías-Martínez, Vijay Karamcheti

Coping with Sparsity in a Recommender System 86
André Bergholz

On the Use of Constrained Associations for Web Log Mining 100
Hui Yang, Srinivasan Parthasarathy

Mining WWW Access Sequence by Matrix Clustering 119
Shigeru Oyanagi, Kazuto Kubota, Akihiko Nakase

Comparing Two Recommender Algorithms with the Help of Recommendations by Peers 137
Andreas Geyer-Schulz, Michael Hahsler

The Impact of Site Structure and User Environment on Session Reconstruction in Web Usage Analysis 159
Bettina Berendt, Bamshad Mobasher, Miki Nakagawa, Myra Spiliopoulou

Author Index 181

LumberJack: Intelligent Discovery and Analysis of Web User Traffic Composition

Ed H. Chi, Adam Rosien, and Jeffrey Heer

PARC (Palo Alto Research Center)
3333 Coyote Hill Road
Palo Alto, CA 94304
{echi,arosien,jheer}@parc.com

Abstract. Web Usage Mining enables new understanding of user goals on the Web. This understanding has broad applications, and traditional mining techniques such as association rules have been used in business applications. We have developed an automated method to directly infer the major groupings of user traffic on a Web site [Heer01]. We do this by utilizing multiple data features of user sessions in a clustering analysis. We have performed an extensive, systematic evaluation of the proposed approach, and have discovered that certain clustering schemes can achieve categorization accuracies as high as 99% [Heer02b]. In this paper, we describe the further development of this work into a prototype service called LumberJack, a push-button analysis system that is both more automated and accurate than past systems.

Keywords: Clustering, Log Analysis, Web Mining, User Profile, User Sessions, World Wide Web

1 Introduction

The Web has become part of the fabric of our society, and accordingly we have an increasing need to understand the activities and goals of Web users. We can improve nearly every aspect of the user experience on a Web site by understanding the users' goal and traffic composition. Webmasters and content producers would like to gain an understanding of the people that are visiting their Web sites in order to better tailor sites to user needs [Yan96]. Marketers would like to know the users' interests in order to have better sale promotions and advertisement placements [Barrett97]. News sites would like to produce and present materials that are highly relevant to their visitors.

Traditional user activity analysis methods such as user surveys are labor-intensive and intrusive when employed daily, slow to apply to on-the-fly Web personalization, and inaccurate due to surveying response inconsistency. Instead, what is needed is an automated means of directly mining the Web server logs for groupings of significant user activities. Web Usage Mining, and more specifically, user session clustering, is a relatively new research area [Cooley97, WEBKDD01, SIAM01], in which user pro

O.R. Zaïane et al. (Eds.): WEBKDD 2002, LNAI 2703, pp. 1–16, 2003.

files are extracted from the server logs and then grouped into common activities such as “product catalog browsing”, “job seeking”, and “financial information gathering”.

Web Usage Mining techniques build user profiles by combining users’ navigation paths with other data features, such as page viewing time, hyperlink structure, and page content [Heer01, Srivastava00]. While the specific techniques vary [Shahabi97, Fu99, Banerjee01, Heer01], the end goal is often the same: to create groupings of user sessions that accurately categorize the sessions according to the users’ information needs.

There are two major issues with existing approaches. First, most approaches examine only one or two combinations of data features for clustering the user sessions. What is needed is an approach that allows for any of the data features to be used, as the situation dictates. For example, sometimes page viewing time might not be available, or page content may be too expensive to gather and analyze, so the user session clustering techniques have to be adaptable to these situations.

Second, in the literature, each technique’s validation is conducted on a different Web site, making it extremely difficult to compare the results. We have no basis from which to choose one data feature over another. What’s worse is that, since only user traces are used, there is no way of knowing *a priori* what the true user information need is for each user session. So we had no way of knowing whether the algorithms performed correctly and clustered the sessions into appropriate groupings.

In this paper, we describe our prototype production system code-named LumberJack that integrates all of this information into a single report that analysts can use in the field to accurately gain an understanding of the overall traffic patterns at a site. We first summarize our previous work on solving both of these issues, and then we describe case studies of LumberJack in action in real-world cases. We also describe challenges and problems that we encountered in practice.

First, we describe our development of a technique that combines any of the available data features to cluster user sessions [Heer01]. The idea of combining multiple features in the context of k-Means clustering is not new [Schuetze99b, Mobasher00, Heer01, Modha02]. The most similar system to ours is described by Mobasher et al. in [Mobasher00]; however, that system does not integrate the two content and usage modalities into a single representation, but instead chooses to cluster the content and usage profiles separately and then choosing the best result from either modality. By integrating these data features into a single system, we can more intelligently pick and choose any of the data features to utilize.

Second, to analyze the effectiveness of the proposed clustering data features, we gathered user sessions for which we know *a priori* the associated information goals, thus enabling us to evaluate whether the clustering algorithms correctly categorized the user sessions [Heer02b]. We present a user study and a systematic evaluation of clustering techniques using these different data features and associated weighting schemes. We first asked users to surf a given site with specific tasks. We then use this *a priori* knowledge to evaluate the different clustering schemes and extract useful guidelines for Web usage analysis.

In the following two sections, we first summarize our clustering approach, and describe the experiment that evaluated the precision of 320 different clustering schemes

[Heer02b]. Next, we discuss LumberJack, an automated web analysis service that combines both user session clustering and traditional statistical traffic analysis techniques. Two case studies illustrating the system's efficacy are presented. Finally, we discuss some of the practical problems encountered and insights gained.

2 Method

The uniqueness of our approach is two-fold. First, our approach to user modeling is motivated by Information Foraging theory [Pirolli99b], and in particular the theoretical notion of Information Scent [Chi00, Chi01]. Information Scent is the user's perception of the value and cost of accessing a piece of information. Applying the notion in this context, we assume implicitly that what a user sees is a part of that user's information interest. Accordingly, we believe that combining the content viewed by the user with their actual navigational path is key to creating models that can accurately infer user's information needs, essentially using information cues to infer user goals [Chi01]. Second, we employ a modular system that combines multiple data features of each Web page, in an approach called Multi-Modal Clustering (MMC), to construct user profiles. Some data feature combinations are novel; for example, past approaches do not use linkage structure in the clustering.

To create the analysis of user traffic, we first model each page in a multi-feature vector space, utilizing page features such as the words, URL, inlinks, and outlinks. This creates a model of all of the pages that users accessed. We then construct a vector space model of user sessions as weighted combinations of the page vectors, using attributes of the users' sessions to determine the weights. We then define a similarity metric for comparing these user sessions and use it to generate the resulting clusters. Fig. 1. below illustrates the steps of this process. We now describe each of these steps in greater detail.

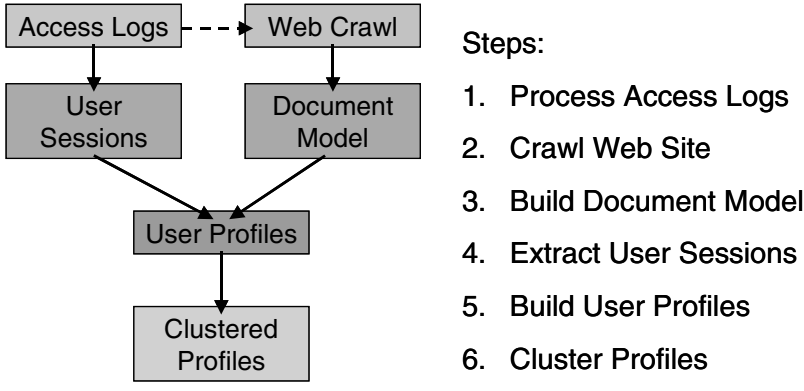


Fig. 1. Method Data Flow–Analysis can be characterized by six stages.

The system begins in **Step 1** by processing web usage logs, typically in the form of standard Web server logs. The logs are initially filtered to extract the relevant URLs. Requests for images and non-existent documents as well as all requests generated by crawlers are removed. In **Step 2**, the content of the site is retrieved using a crawler. Any relevant pages listed within the access logs that were not captured by the basic crawl are retrieved, allowing the system to capture dynamically generated pages such as search results.

In **Step 3**, the retrieved content is used to build a vector space model of the pages on the site. The system currently models four feature spaces in its representation: Content, URL, Outlinks, and Inlinks. Each feature space is called a *modality*. While these modalities are not exhaustive, they do provide good coverage of the textual and navigational content of the site. For example, one can imagine other interesting possibilities like visual page features such as color histogram data, banner ad placements, and navigational structures. We now describe each modality in more detail:

Content: The **Content** subvector of a page is a weighted keyword vector modeling the words that the user sees on that page. The words visible to the user are extracted, stop words are removed, and the remaining words are run through Porter's stemming algorithm [Porter80].

URL: The **URL** subvector of a page is a URL token keyword vector modeling the URL of the page. Each URL is tokenized using '/', '&', '?' and other appropriate delimiters.

Inlink/Outlink: The **Outlink** subvector of a page describes which pages are reachable from this page, while the **Inlink** subvector describes which pages link to this page.

These individual subvectors are then concatenated to form a single multi-modal vector $\boldsymbol{P}_d = (\boldsymbol{Content}_d, \boldsymbol{URL}_d, \boldsymbol{Inlink}_d, \boldsymbol{Outlink}_d)$ for each document d. The entire vector is represented in sparse vector format. Of course, in practice we can always pick and choose which modalities we want to include in the model.

We also selectively apply the Term Frequency by Inverse Document Frequency (TF.IDF) weighting scheme on the modalities. A common technique in the information retrieval field, TF.IDF provides a numerical value for each item in a document, indicating the relative importance of that item in the document. This weighting is roughly equal to an item's frequency in a given document divided by the frequency of the item occurring in all documents. Formally, TF.IDF can be expressed as $TF.IDF_{t,d} = \log(1 + tf_{t,d}) * \log(N / df_t)$, where $tf_{t,d}$ indicates the number of times an item t occurs in a given document d, and df_t indicates the number of documents in which the item t appears, and N indicates the total number of documents [Scheutze99a, p. 542]. In practice, we have found that TF.IDF works nicely in the content and URL modalities, and poorly in the linkage modalities.

In **Step 4**, we construct the user session model by representing the individual user sessions as vectors. We first sessionize the usage logs using standard techniques [Pirolli99a] and then represent each session as a vector $\boldsymbol{s}$ that describes the sessions' page

views. We have explored a number of possibilities for weighting the pages in a session ***s***. These **path weightings** possibilities are:

(a) Frequency (or Uniform): Each page receives weighting equal to the number of times it is accessed, e.g. for a user ***i,*** A→B→D→B, $\boldsymbol{s}_i$ = (1,2,0,1,0).

(b) TF.IDF: Treating each session as a document and the accessed pages as the document terms, each page receives a TF.IDF weighting.

(c) Linear Order (or Position): The order of page accesses in the session is used to weight the pages, e.g. A→B→D **,** $\boldsymbol{s}_i$ = (1,2,0,3,0).

(d) View Time: Each page in the session is weighted by the amount of viewing time spent on that page during the session, e.g. A(10s) → B(20s) → D(15s), $\boldsymbol{s}_i$ = (10,20,0,15,0).

(e) Various Combined Weighting: Each page in the session is weighted with various combinations of the TF.IDF, Linear Order, and/or View Time path weighting. For example, using both Linear Order+View Time: A(10s) → B(20s) → D(15s), $\boldsymbol{s}_i$ = (10,40,0,45,0).

In **Step 5**, we combine the document and session models to construct a set of user profiles. We assume implicitly that each page a user sees is a part of that user's information interest. Each user profile $\boldsymbol{UP}_i$ is constructed as the linear combination of the page vectors $\boldsymbol{P}_d$ scaled by the weights in $\boldsymbol{s}_i$. That is,

$$UP_i = \sum_{d=1}^{N} s_{id} P_d$$

In order to do comparison between user profiles, we need to define a similarity metric ***d()*** over the profiles. Our system uses the standard cosine measure, which measures the cosine of the angle between two vectors, applied to each modality individually. The values of these comparisons are then linearly combined to obtain the resulting similarity value. Because we use the cosine measure, each user profile must first undergo normalization, where each modality subvector is normalized to unit length. We can formally represent the similarity measure as:

$$d(UP_i, UP_j) = \sum_{m \in Modalites} w_m \cos(UP_i^m, UP_j^m) \qquad \sum_m w_m = 1$$

By adjusting the **modality weights** w_m, we can specify the relative contribution of each modality in the similarity function, for example by weighting page content higher than the other modalities.

Finally, in **Step 6**, using this similarity function we can then proceed with clustering. We use a recursive bisection approach as described in [Zhao01], which starts by placing all user profiles in one initial cluster and then repeatedly bisects clusters using the traditional K-Means algorithm [MacQueen67] until a specified number of clusters *k* is achieved. The corresponding criterion function maximizes the within cluster similarity between members of a cluster *S* and the cluster's centroid *C*:

$$\max \sum_{r=1}^{k} \sum_{UP_i \in S_r} d(UP_i, C_r)$$

This criterion has proven very successful within the context of document clustering [Zhao01], and so we apply it here with high expectations, as our user profiles are in fact document aggregates. Though not reported here, we have indeed found this criterion to work best among a number of clustering techniques. Details of this specific approach can be found in [Zhao01].

In previous work [Heer02a], we have explored using a stability-based method [BenHur02] for finding the optimal number of clusters. Though the method displayed some promise, overall our results proved inconclusive. As we collect more usage data sets in the future, we hope to settle on an acceptable scheme.

In summary, we model both the web site and the collected usage data, and then combine these models to generate a set of user profiles represented as multi-modal vectors. We then cluster these profiles to obtain categorizations of the user sessions.

3 Evaluation

We recently performed a systematic evaluation of different clustering schemes using the above approach by conducting a user study where we asked users to surf a large corporate site with *a priori* specified tasks [Heer02b]. That is, we specify to the users what their goals should be, and ask them to surf according to that user intent. By knowing what the tasks were and how they should be grouped in advance, we were able to do post-hoc analysis of the effectiveness of different clustering schemes. Here we describe this experiment briefly.

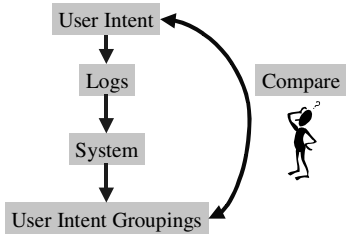


Fig. 2. By conducting a user study where we give the users specific tasks, we know *a priori* how the tasks should have been grouped. In the experiment, we run the user traces through the system, and compare the results of the system analysis with the *a priori* groupings.

We asked 21 participants to surf Xerox.com and captured their clickstreams using the WebQuilt proxy [Hong01]. Each user performed a total of 15 tasks, organized into five information need groupings consisting of three tasks each. Individual tasks were chosen using email feedback from the site. The five task groups were “product info”,

"support", "supplies", "company info", and "jobs". We assigned the tasks randomly, but with each task assigned roughly the same number of times. Subjects were asked to perform the tasks in their natural browsing environment and allowed to start and stop a task anytime they wanted. We collected a total of 104 user sessions.

We studied a total of 320 different algorithm schemes using various modality weighting and path weighting methods in combination. Some modules would be turned on, such as TF.IDF, during certain schemes. As depicted in Fig. 1., we then measured accuracy by counting the number of correct classifications (comparing against our *a priori* task categories) and then dividing by the total number of sessions.

We discovered that, by counting the number of correct categorizations, certain combinations of data features enabled us to obtain accuracies of up to 99%. The traditional scheme of clustering just the user session vectors (denoted in the following figures as "Raw Path") gives an accuracy of only 74%, while certain combinations give accuracies below 60%.

Two trends were significant. First, Content performed admirably, with poor performance only when used with the TF.IDF path weighting alone. Fig. 3. shows how content outperformed other uni-modal schemes. A Linear Contrast showed that content performed significantly better than the other four uni-modal schemes ($F(1,35)=33.36$, $MSE=.007332$, $p<0.0001$). Expanding this to all multi-modal schemes, we compared all of the content-based schemes vs. non-content-based schemes. This contrast was also significant ($F(1,105)=32.51$, $MSE=.005361$, $p<0.0001$). Thus, crawling the site and using the page content to help cluster user sessions greatly increases algorithm accuracy. This is far from surprising; intuitively, the words that the user sees during each session are good indicators of their information need.

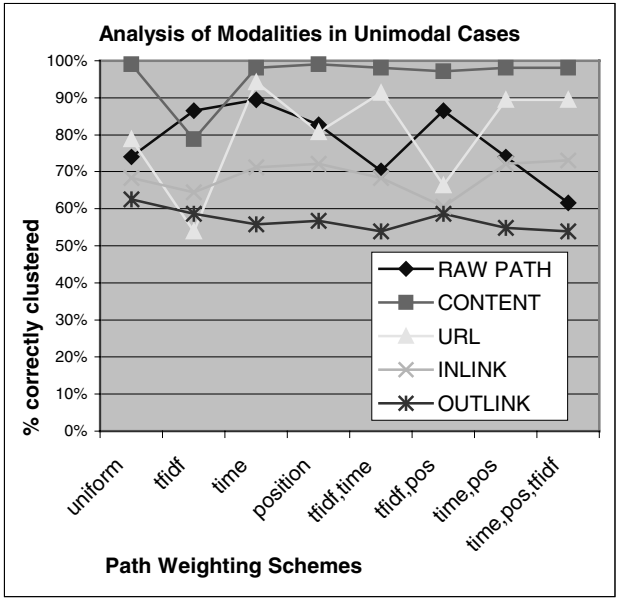


Fig. 3. (a) Plot of each different modality's accuracies across different path weighting schemes.

Second, we found that the viewing time spent at each page is a good data feature for clustering the user sessions. Fig. 4 shows the analysis of path weighting. The left portion of the chart shows the uni-modal cases, while the right side shows the multi-modal cases. "Raw Path" in the Figure denotes clustering just the user page-view vectors, which is the traditional Web usage clustering method. What's most interesting from this chart is that the View Time path weighting performed well, staying at the top of the curve across the chart. This is true regardless whether we're looking at the uni-modal or the multi-modal schemes. A paired t-Test found a significant difference between View-Time-based schemes vs. non-View-Time-based schemes (n=60, *V.T.mean*=89.5%, *s.d.*=12.7%, *non-V.T.mean*=83.2% *s.d.*=12.0%, *t(59)*=4.85, *p*=4.68e-6).

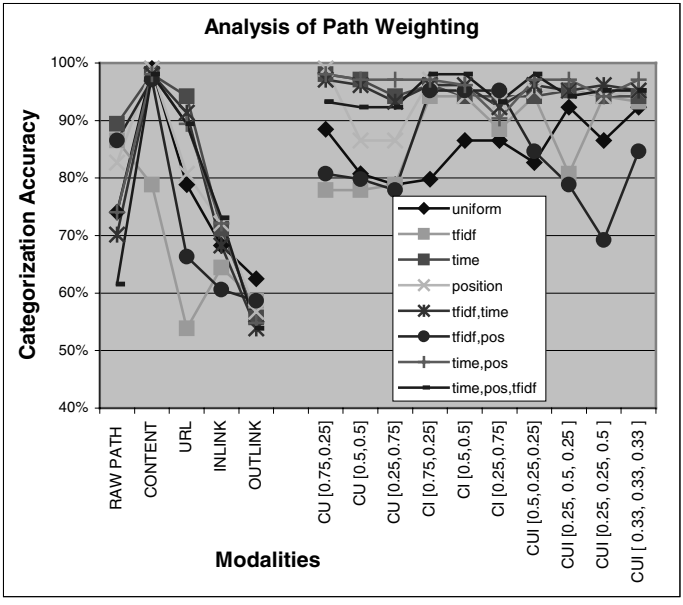


Fig. 4. Plot of each path weighting schemes against modality. (Modality shorthands are C=Content, U=URL, I=Inlink, O=Outlink)

We also discovered that multi-modal schemes are more robust in real life applications. There are many Web sites that have pages containing only images, sounds, videos, or other media formats. These pages cannot be parsed for content, making the usage of other modalities much more important. We could tailor the clustering technique to the unique features of the site being analyzed. For example, if the site has many pages without word content, then Inlink and URL Token modalities could be used in combination with View Time.

4 LumberJack Log Analyzer

Encouraged by the successful evaluation, we applied our approach to a new service, code-named LumberJack, which is designed to remove the responsibility of the analyst to "chunk" behaviors, creating a robust starting point for the analyst to begin understanding the behavior of each type of user. Using the techniques above, LumberJack processes server logs and automatically crawls the content and hyperlinks of the site to construct a model of user activity. It performs the clustering analysis described previously (in part using George Karypis' CLUTO toolkit [CLUTO02]), and then computes a number of statistical analyses for *each* discovered user group, in contrast to statistics about the entire population shown in typical user activity reports. LumberJack is designed to be a push-button operation, in that only the server logs are needed, and the rest of the system runs completely automatically.

Table 1. Summary of LumberJack Report Details

Report detail	Statistics	Benefit
User group summary	For each group: Number of sessions, percentage of population, Keywords that best describe the user group, Cluster quality (internal/external similarity)	Segments users by need/activity
Most frequent documents	Top ten most frequently occurring documents with absolute and relative time spent.	Where are users going? Are users focused on key pages, or do they have broad interests?
Representative paths	Sequences of pages that best describe typical activity (user sessions nearest cluster centroid)	Reduces session population to show best examples
Session path length trends	Histogram of user session path lengths	Characterizes "Law of Surfing" behavior: Are most paths long or short?
Session time trends	Histogram of user session times	Characterizes "Law of Surfing" behavior: Do most sessions include long reading times?
Session Document Co-occurrence	How often the most frequent documents appear in the same session for all sessions?	Shows task success rates
Session Document Transitions	How often the most frequent documents are followed by another in a session? How often each document is the first or last document in a session?	Characterizes attrition rates between pages. Detect "pogosticking", i.e., repeated sequences of forward/backward transitions.

One motivation for LumberJack has been a lack of robust identification of user needs in traditional traffic analyses. For example, we might know that users travel from a product list to a specific product 12% of the time, but don't know under what specific circumstances this aggregate behavior occurs; perhaps there are two classes of users, one that knows exactly what type of product they want, the other who are only interested in browsing many products to understand the available options. An analyst is often required to infer these circumstances herself based upon little or no information. They may sort through data manually or make guesses, but user session clustering provide a much better foundation an analyst may rely upon. Conversely, solely discovering user groups is not necessarily actionable for an analyst either. Analyses of user needs and traffic are most useful when paired together.

The LumberJack output is a Web-based report, describing the different user groups and their properties. Table 1 summarizes the various cluster statistics available.

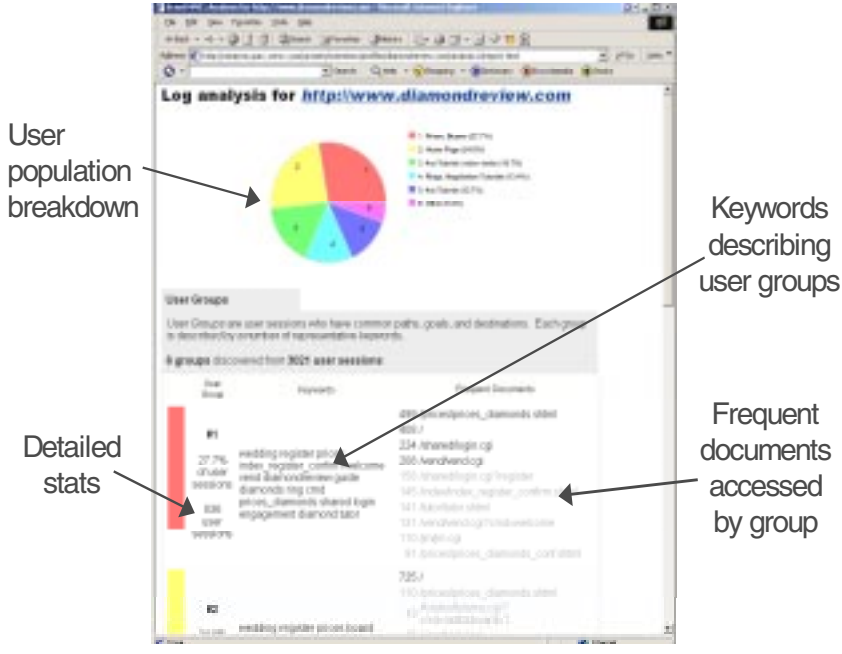


Fig. 5. Sample LumberJack report in *simple* mode for DiamondReview.com, Jan. 4-8, 2002

Fig. 5. illustrates a sample LumberJack report for a consumer information web site about diamonds. The current system generates static reports for a pre-specified number of clusters. In the future we plan to extend the reports to support interactive exploration of the generated cluster hierarchy, allowing analysts to refine their focus to particular clusters of interest, viewing user goals at multiple levels of granularity.

When used to analyze the site DiamondReview.com (Fig. 5), LumberJack discovered six user groups in 3021 user sessions for 4 days starting from Jan. 4, 2002. 27.7% of the users were buyers interested in prices, while 24.0% spent most of their

time on the home page. 16.7% were return visitors going through a diamond tutorial, and we know this because they had obviously started the tutorial from a half-way point that they have bookmarked. 13.4% were ring buyers, and 12.7% were first time tutorial readers. The remaining 5.4% were users that looked at many different pieces of information. Additionally, of the return tutorial users, the median session length was 11 clicks, mid-way into the tutorial, and at each step of the tutorial, the attrition rate was only 10%. It was around 35% for first time tutorial viewers; we hypothesize that those users who returned would have a greater impetus to continue the tutorial and finally making a purchasing decision.

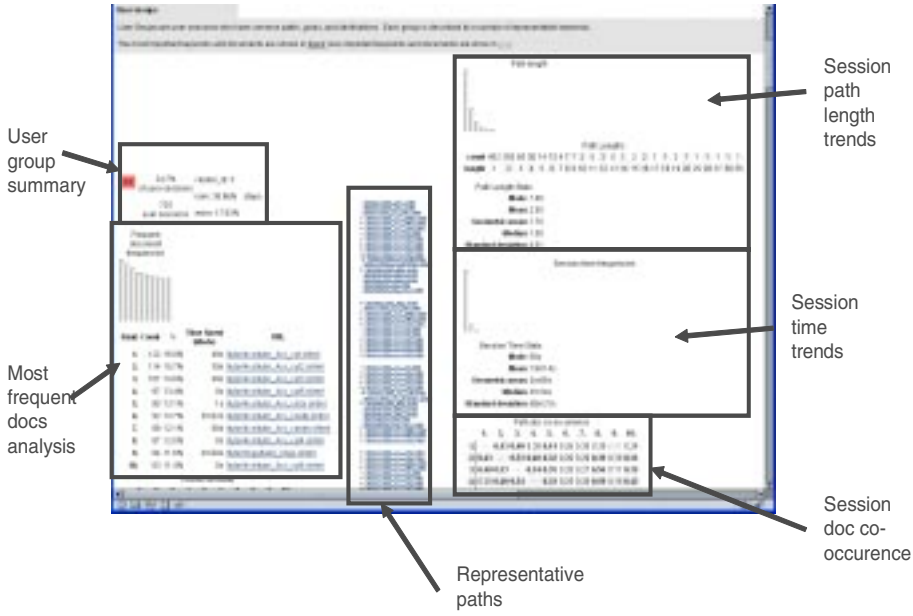


Fig. 6. Sample LumberJack report in *detailed* mode for DiamondReview.com.

Our report directly impacted the company in one significant way. We discovered that most users that starts the diamond tutorial finish reading the tutorial, even though it takes 30 minutes to finish going through the entire tutorial eventually. The designers of the site had anticipated that most users would not choose to finish the tutorial, so a link to diamond prices was placed at every step of the tutorial as an exit point, except on the last page. Since most users finish the tutorial, we recommended to Diamondreview to include purchasing information at the end of the tutorial.

We also studied using our tool on very large corporate sites. Fig. 6. shows the result of running this tool on the logs from Xerox.com on July 24, 2001. The most striking result is that a large segment of user sessions (41.4%) center around the splash page. Viewing the actual clustered sessions revealed that these sessions consisted

primarily of the site's splash page, with many paths jumping between the splash page and other linked pages. This could indicate that a large segment of users may come to the site without well-defined information needs and/or that the site may suffer possible usability problems that prevent users from successfully moving deeper into the site.

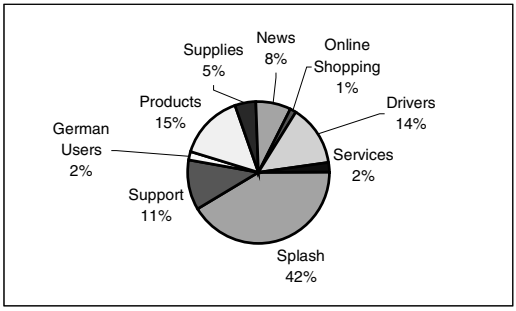


Fig. 7. User Groups discovered for Xerox.com, July 24, 2001

Other substantial groupings include Xerox Product Catalog browsing (15%), Driver downloads (13.9%), Technical Support (11.5%), and Company News and Information (8.2%). One unexpected result was that there was a strong, concentrated group of German users that necessitated a unique cluster (1.7%). Xerox Sales and Marketing might also be interested to know the number of Online Shopping / Purchase related sessions (1.3%) in comparison to the number of product catalog viewers. More detailed information about these groupings can be obtained by reclustering a given cluster [Heer01]. For example, we learned that within the Products group, 43% of the sessions centered around the Phaser line of printers. This information could be extremely useful to the marketing department, who can more carefully place advertisements of the Phaser printers at strategic routes common to the Products group.

In this section, we have presented our log analysis tool, LumberJack, which automatically analyzes the user traces obtained from a server log and clusters user sessions into significant groupings that describe the traffic composition at a site. It generates reports describing the properties of these groupings, helping analysts understand the various activities of users on their sites. These case studies illustrate that these user session clustering methods are indeed applicable to real world situations, and are scalable to large, heavily used Websites.

5 Difficulties and Lessons Learned

While developing our analysis techniques and building the LumberJack system, we have encountered a number of practical issues. First, to obtain the content modality, crawling and parsing a wide variety of sites currently presents the greatest challenges. For example, sites may dynamically create pages with unrepeatable URLs that cannot be retrieved later, or pages containing dynamic links generated by JavaScript. Sometimes breadth-first traversals of a site (typical of most crawlers) may not be possible

because of business logic requiring sequences of pages to be accessed in a particular order. Indeed, we have encountered many other technical issues.

However, LumberJack can compensate for the lack of any one data modality, because the multi-modal nature of LumberJack's user profiles provides this robustness. For example, if links cannot be resolved to valid URLs of a crawl, one may trade the use of the inlinks modality for another. The necessity of trade-offs will hopefully be reduced in the future by increased technical sophistication of crawlers and parsers, greater standardization of site designs and logic, or more clever solutions such as the instrumentation of proxy servers. Moreover, the choices of which modalities to use may be automated by approaches similar to Caruana and Freitag's work on using hillclimbing to do feature selection [Caruana94]. They were able to obtain good results in Mitchell's Calendar APprentice Project.

Second, in practice, the high dimensionality of the multi-modal vectors must be represented very efficiency using sparse vector format. We use the existing compaction techniques for this purpose. Also, while we were able to obtain accurate results using k-Means in our laboratory studies and in real-world situations, we are not entirely certain that this method can tackle all of the complex sites out there with varying user tasks. Further studies are needed to understand the applicability in a wide variety of situations. Existing literature suggest that k-Means is often unable to tackle complex high dimensional spaces with low cluster separations. However, since our use of k-Means is only limited to doing bisections, this problem is minimized somewhat. We suspect that with complex sites a graph partitioning method might be more appropriate and more robust.

Third, our current work contains no evaluation of the sessionization technology used during usage clustering. Our user study evaluation obtained user trace data that is perfectly sessionized. Data from the real world is unlikely to contain user sessions with multiple information goals, for example. Unless the same kind of information goals co-occur frequently (e.g. look up stock data and company executive team), it is unlikely that these sessions can be easily clustered, potentially requiring sessions to belong in multiple clusters for example.

Most importantly, while our algorithm remains to be only log analysis algorithm that is truly validated with a user study with *a priori* defined tasks, we would like to obtain data to validate it against many other sites to fully understand the algorithm's robustness. Our experience with real-world log files from real sites thus far has given us confidence that the algorithm performs well by using content and view-time as the primary parameters. But because our evaluation is currently done using a single site, we do not know the robustness of the algorithm across a wide variety of different sites. This kind of expanded study should shed light on how the modalities should be used during clustering.

Finally, our unique approach to the problem is deeply rooted in our user-centered point of view. We have designed the mining method from the ground-up to use the information cues that a user encounters as she surfs from one page to another. We use these information cues to try and understand how this conveys her information goals. Our experimental study is also rooted in this unique approach by evaluating it using user data that we know is as clean as possible. We have written extensively on this

Information Scent approach elsewhere [Chi00, Chi01]. We believe that, in this domain, approaches and studies should always be done using this user-centric approach.

6 Conclusion

As analysts, we are deeply involved in making the Web more accessible to users. We need to know what users are doing in order to better optimize the Web sites. Recent research seeks to understand the composition of user traffic using Web Mining techniques on server logs. The commonality is to first build up user profiles based on the user visitation paths, and then apply clustering techniques to these user profiles.

This paper summarizes several years of work. First, we presented a framework in which any of the available data features can be used in the clustering of user sessions. This framework enables robust combinations of content, linkage structure, and usage to discover user needs. Second, we presented results from a systematic evaluation of different clustering schemes by conducting a user study where we asked users to surf a large corporate site with *a priori* specified tasks. By knowing what the tasks were and how they should be grouped in advance, we were able to do post-hoc analysis of the effectiveness of different clustering schemes. We discovered that, by counting the number of correct categorizations, certain combinations of clustering schemes enabled us to obtain accuracies of up to 99%. While we don't necessarily expect this same level of accuracy on real world web logs, which are much noisier by nature, this is still quite encouraging to analysts hoping to make sense of user actions on the web.

We have since taken this knowledge and built LumberJack, a prototype automated analysis system that couples user session clustering with more traditional statistical analyses of web traffic. This prototype uses a set of preset parameters from this study to obtain essentially a push-button tool, paying particular attention to content and view-time as its primary data modalities to analyze. This combination seeks to not only identify significant user information goals, but also to understand how users follow these goals as they surf the site. The goal is to create a completely automatic system that quickly and accurately identifies a site's dominant usage patterns.

In the quest to understand the chain of user interactions on the Web, analysts need tools for getting quick and accurate pictures of Web usage. The work presented here suggests that in many cases the structure of user activity may be inferred automatically.

Acknowledgements. The authors would like to thank the comments and insights of George Karypis and Peter Pirolli. Pam Desmond helped with proofreading. This work was supported in part by Office of Naval Research grant No. N00014-96-C-0097 to Peter Pirolli and Stuart Card. Finally, we would like to thank the sites who have donated their logs for analysis and all the subjects who participated in our evaluation.

References

[Banerjee01] Banerjee, A. and Ghosh, J. Clickstream Clustering using Weighted Longest Common Subsequences, in *Proc. of the Workshop on Web Mining, SIAM Conference on Data Mining* (Chicago IL, April 2001), 33–40.

[Barrett97] Barrett, R., Maglio, P.P., and Kellem, D.C. How to personalize the Web, in *Proc. of the ACM Conference on Human Factors in Computing Systems, CHI '97* (Atlanta GA, March 1997), 75–82.

[BenHur02] Ben-Hur, A., Elisseeff, A., and Guyon, I. A Stability Based Method for Discovering Structure in Clustered Data, in *Proceedings of the Pacific Symposium on Biocomputing (PSB2002)*, January 2002, Kaua'i, HI.

[Caruana94] Caruana, R., and Freitag, D. 1994. Greedy attribute selection. In *Proc.of International Conference on Machine Learning, ML-94*, pp. 28–36. Morgan Kaufmann.

[Chi00] Chi, Ed H., Pirolli, P., and Pitkow, J. The Scent of a Site: A System for Analyzing and Predicting Information Scent, Usage, and Usability of a Web Site. In *Proc. of ACM CHI 2000 Conference on Human Factors in Computing Systems*, pp. 161–168, 581, 582. ACM Press, 2000. Amsterdam, Netherlands.

[Chi01] Chi, E.H., Pirolli, P., Chen, K., and Pitkow, J. (2001). Using information scent to model user information needs and actions on the Web. *Proc. of the ACM Conference on Human Factors in Computing Systems, CHI 2001* (pp. 490–497), Seattle, WA.

[Cooley97] Cooley, R., Mobasher, B. and Srivastava, J. Web Mining: Information and Pattern Discovery on the World Wide Web. In *Proc. of the International Conference on Tools ith Artificial Ingelligence*, pp. 558–567. IEEE, 1997.

[CLUTO02] CLUTO: A Software Package for Clustering High-Dimensional Datasets. Available at http://www-users.cs.umn.edu/~karypis/cluto/

[Fu99] Fu, Y., Sandhu, K., Shih, M. Asho Generalization-Based Approach to Clustering of Web Usage Sessions, in *Proc. of WEBKDD 1999* (San Diego CA, August 1999), 21–38.

[Heer01] Heer, J. and Chi, E.H. Identification of Web User Traffic Composition using Multi-Modal Clustering and Information Scent, in *Proc. of the Workshop on Web Mining, SIAM Conference on Data Mining* (Chicago IL, April 2001), 51–58.

[Heer02a] Heer, J. and Chi, E.H. Mining the Structure of User Activity using Cluster Stability, in *Proceedings of the Workshop on Web Analytics, SIAM Conference on Data Mining* (Arlington VA, April 2002).

[Heer02b] Heer, J. and Chi, E.H. Separating the Swarm: Categorization Methods for User Access Sessions on the Web. In *Proc. of ACM CHI 2002 Conference on Human Factors in Computing Systems*, pp. 243–250. ACM Press, April 2002. Minneapolis, MN.

[Hong01] Hong, J.I., Heer, J., Waterson, S., and Landay, J.A. WebQuilt: A Proxy-based Approach to Remote Web Usability Testing, to appear in *ACM Transactions on Information Systems*.

[MacQueen67] MacQueen, J. Some methods for classification and analysis of multivariate observations, in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1 (1967), UC Berkeley Press, 281–297.

[Mobasher00] Mobasher B., Dai, H., Luo, T., Su, Y., and Zhu, J. Integrating usage and content mining for more effective personalization. In *Proceedings of the International Conference*

on E-Commerce and Web Technologies (ECWeb2000), *Lecture Notes in Computer Science 1875* (Sept. 2000). Springer Verlag,.

[Modha02] Modha, D. and Spangler, W. Feature Weighting in k-Means Clustering. *Machine Learning*, 47, 2002.

[Porter80] Porter, M.F., 1980, An algorithm for suffix stripping, *Program*, 14(3) :130–137

[Pirolli99a] Pirolli, P. and Pitkow, J.E. Distributions of Surfers' Paths Through the World Wide Web: Empirical Characterization. *World Wide Web*, 2(1–2), 1999. 29–45.

[Pirolli99b] Pirolli, P. and Card, S. K. (1999). Information Foraging. *Psychological Review* 106(4): 643–675.

[Schuetze99] Schuetze, H. and Manning, C. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge MA, 1999.

[Schuetze99b] Schuetze, Hinrich, Pirolli, Peter, Pitkow, James, Chen, Francine, Chi, Ed, Li, Jun. System and Method for clustering data objects in a collection. Xerox PARC UIR QCA Technical Report, 1999.

[Shahabi97] Shahabi, C., Zarkesh, A.M., Adibi, J., and Shah, V. Knowledge Discovery from User's Web-page Navigation, in *Proc. 7th IEEE Intl. Conf. On Research Issues in Data Engineering* (1997), 20–29.

[SIAM01] *Proc. of the Workshop on Web Mining, SIAM Conference on Data Mining* (Chicago IL, April 2001).

[Srivastava00] Srivastava, J., Cooley, R., and Deshpande, M. (2000) Web Usage Mining: Discovery and Application of Usage Patterns from Web Data. *SIGKDD Explorations* 1(2): 12–23.

[WEBKDD01] *Proc. of the SIGKDD Workshop on Web Data Mining (WEBKDD01)* (San Francisco CA, August 2001).

[Yan96] Yan, T.W., Jacobsen, M., Garcia-Molina, H., and Dayal, U. (1996), From User Access Patterns to Dynamic Hypertext Linking. *Computer Networks*, vol. 28, no. 7–11 (May 1996), 1007–1014.

[Zhao01] Zhao, Y. and Karypis, G. (2001). Criterion Functions for Document Clustering: Experiments and Analysis. Technical Report #01–40. University of Minnesota, Computer Science Department. Minneapolis, MN.

Mining eBay: Bidding Strategies and Shill Detection*

Harshit S. Shah, Neeraj R. Joshi, Ashish Sureka, and Peter R. Wurman**

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7535 USA
harshit@shah-family.net
neerajrj@yahoo.com
asureka@unity.ncsu.edu
wurman@ncsu.edu

Abstract. Millions of people participate in online auctions on websites such as eBay. The data available in these public markets offer interesting opportunities to study Internet auctions. We explore techniques for identifying common bidding patterns on eBay using data from eBay video game console auctions. The analysis reveals that there are certain bidding behaviors that appear frequently in the data, some of which have been previously identified and others which are new. We propose new attributes of bidding engagements and rules for classifying strategies. In addition, we suggest economic motivations that might lead to the identified behaviors. We then apply a clustering algorithm to look at each bidder's behavior across several engagements, and find a few natural clusters. Finally, we apply association rule analysis to the data and find cases of likely shill behavior, but find no cross-correlation with any particular bidding behavior.

1 Introduction

Once considered esoteric by the general public, auctions now engage millions of people daily. At any given time, there are millions of auction listings across thousands of categories on auction sites such as eBay, Yahoo, and uBid.

Auction sites rank high in both the number of visitors and the average time spent per visit, and there are myriad reasons to try to better understand how users interact with the service. From the system architecture point of view, it is important to know how the users' actions are distributed over the life of the auction. A market designer would like to know how the choice of auction rules affect the load on the server. Bidders may be able to use this information to improve their individual bidding strategies and build intelligent software agents to support their economic activities, while sellers can use this information to improve their revenue. Economists may find the information valuable as they analyze the performance of these auction sites as social institutions. Finally, an understanding of normal and abnormal bidding behavior can help authorities track down fraud.

* This paper is an extended version of a paper presented at WEBKDD 2002[1].
** Contact author.

O.R. Zaïane et al. (Eds.): WEBKDD 2002, LNAI 2703, pp. 17–34, 2003.

Data mining techniques can help these various interested parties to sift through the vast amounts of data generated in online auctions. Although there are potentially many benefits from mining bidding data, the domain also poses some particular challenges. First, the interactions between the bidder and the auction system have many attributes (time, value, number of bids, reputation, product description, etc.), some of which are measured only imprecisely, and some of which are implicit, but useful, characterizations of the data. In addition, the data has a temporal dependency that may be linked not only with the internal clock of the auction, but also, externally, to the greater economy. Finally, the data tends to be exceedingly sparse, as will become evident below.

In this work, we examine the actual behavior of bidders on eBay with the goal of trying to answer the following specific questions:

1. Is it feasible to classify the bidding behavior of individual bidders?
2. If so, what strategies are common on eBay?
3. Can we identify enough bidders to make it worthwhile?
4. Can we detect suspicious fraudulent behavior?

In order to address these questions, we accumulated bidding data from nearly 12,000 completed eBay auctions. In Section 2, we describe eBay's auction mechanism and it's salient features. In Section 3, we discuss how the data was collected and interpreted, with particular emphasis on reverse engineering the ask prices after each bid. We analyze various bidding strategies in Section 4, and attempt to classify bidders based on the strategies. Section 5 uses the same data to mine for suspected fraudulent behavior, with some surprisingly strong results. Section 6 discusses related work. The last section includes a discussion of the implications of this work and some of the potential future directions.

2 eBay – Model and Mechanism

All eBay auctions use an ascending-bid (English) format with the important distinction that there is a fixed end time set by the seller.[1] Ebay provides a standard auction and three variations:

Standard Auction: This is the most prominent type of listing. Here only one item (or group of items sold together) is being offered to the highest bidder.

Reserve Price Auction: The seller has a hidden reserve price that must be exceeded before the seller is required to sell. The first time a buyer places a bid equal to or greater than the reserve price, the item's current price is raised to the reserve price amount.

Buy It Now Price: A variation of the standard auction in which a bidder can immediately win the item by choosing the Buy It Now option. If selected by the seller, this option is available until the first bid (or the first high bid that meets or beats the reserve price). A single item auction ends prematurely once a bidder exercises this option.

[1] Other online sites may differ from this approach by providing a flexible end time for the auctions, which will greatly impact the bidders' strategies [2].

Dutch Auction: The seller offers more than one of the exact same items. The bidder enters the quantity of the items desired along with the price he is willing to pay per item. All winners pay the lowest winning bid price.

Regardless of the auction type, eBay uses a proxy mechanism for all submitted bids. The proxy mechanism allows a bidder to submit a *maximum bid* (i.e., maximum willingness to pay) with a guarantee that eBay will raise the bidder's active offer automatically until the bidder's maximum bid value is reached. We refer to the bid placed by the proxy system as the bidder's *proxy bid*. In a reserve price auction, the seller's reserve price is treated like any other bid; if the buyer's offer meets or exceeds the reserve (secret) bid set by the seller, the buyer's bid would be raised to that price immediately. If no buyer exceeds the reserve, the seller "wins" the object.

Ebay enforces a minimum bid increment that, along with the current ask price, determines a lower bound on bids the server will accept. The bid increment table specified by eBay defines a schedule in which the increments increase as the current ask price increases.[2]

3 Data Collection and Interpretation

We collected data for the auctions of Sony Playstation 2 console (PS2) and Nintendo Gameboy Advanced consoles (GBA) around their respective product release dates. We choose these product categories because, during the periods in question, supply greatly lagged demand. The products could be purchased in retail outlets for a fixed price, but because buyers had private values significantly greater than the retail value at the time when the data was collected, a liquid secondary market instantly developed on eBay. In addition, it is reasonable to assume that a normal consumer is likely to need no more than one of either product, and that the products are purchased for use and not as an investment.

The PS2 data was collected over a two-day period in October 2000 (PS2 launch in US) and for 3 weeks during January 2001. The GBA data was collected from May 31 to July 29, 2001 (GBA launch in US - June 11, 2001). In total, details of 11,537 auctions were collected. All of the data has been anonymized.

Ebay keeps the data from completed auctions available on its website for the most recent thirty days. To collect the data, we wrote a spider that executes a search through the historical data for each product category. From the search results, the spider constructs the URLs to request individual auction data and the bidding history pages. The bid history page contains the details of all of the bids submitted to the auction. The spider caches both the auction details and bid history pages for each completed auction and parses them later. All requests are staggered to avoid putting a noticable load on eBay's server.

From the cached pages, the auction details were extracted and stored in a database. We refer to the complete set of data as data set D. The bid history pages show the time, bid value (e.g., maximum bid),[3] and bidder ID for each bid placed in the auction.

[2] Ebay Help System - http://pages.ebay.com/help/basics/g-bid-increment.html.

[3] More precisely, we know the maximum bid of every bidder except the one who won the auction. For the winner, eBay records a bid that is one bid increment above the second highest bidder.

Reconstructing the flow of the auction, however, is a subtle process because the bid history page does not show us the proxy bids that were placed by eBay's system on behalf of the bidders. Thus, the *ask-price* (e.g., the price that eBay announced on its web page) that the bidder saw when submitting his bid is not recorded. In addition, in reserve price auctions, the reserve price cannot be recovered from the data supplied. Similar challenges apply to analyzing the Buy-it-now enabled auctions and the multi-unit Dutch auctions. Thus, for much of the analysis we restrict attention to standard auctions, which account for 60% of the data set. We refer to this restricted data set as D_r.

To make the discussion more formal, consider an auction, k, on eBay. Let $j \in J_k \subseteq J$, where J is the set of all bidders that appear in the restricted data set D_r and J_k represents the subset of bidders who participate in auction k. Let B_j represent the set of bids placed by bidder j in the auction, and denote the i^{th} member of the set as b_{ij}. We refer to B_j as an *engagement*.

We define the time range of the auction as $t_s - t_{end}$, where t_s and t_{end} represent the start and end time of the auction, respectively. At any time t, such that $t_s \leq t \leq t_{end}$, eBay announces an ask price, denoted by π_t. The minimum bid at time t, denoted b_t is:

$$b_t = \pi_t + \theta(\pi_t)$$

where $\theta(\pi_t)$ is the bid increment from eBay's minimum bid increment table.

Clearly there are attributes of the bid that are explicit in the data and which may be informative, such as the value of b_{ij} and the time at which it was placed. We propose that the *context* of the bid be used also to infer strategy, especially the difference between the actual bid value and the minimum bid required by eBay. To compute this, we introduce a derived parameter in the model. The *excess increment* of a bid is defined as the excess amount over the minimum bid, and represents the bidder's use of the proxy system. We denote the excess increment of bid b_{ij} as δ_{ij}, and calculate it by:

$$\delta_{ij} = b_{ij} - b_t = b_{ij} - \pi_t - \theta(\pi_t).$$

However, to compute the excess increment requires that we know the ask-price (π_t) at the time b_{ij} was placed. It is possible, but non-trivial, to recover π_t from the data available on eBay. From the bid history, we can calculate the ask prices, and then the excess increment of all bids submitted to the auction, with the exception of the winning bid.

To calculate the ask price and excess increment of the bids for an auction, we replay the auction by sorting the bids according to bid-time and then processing each bid one-by-one in the manner eBay would. In the process, we calculate the excess increment of the bid, the new ask price, and the high bidder. Unfortunately, there is no official publication provided by eBay that precisely describes the bid processing algorithms. We discovered the process by reverse engineering from the data using the descriptions on eBay's Help System, and our own experience developing auction systems, with some details worked out by trial-and-error.

The procedure for admitting bids is as follows:

i. The starting price is the minimum initial bid set by the seller.
ii. The first bid does not affect the current ask price, but the bidder becomes the high bidder at the starting price.
iii. Any new bid must at least match the minimum bid at that time.[4]
iv. When a bid is admitted, the new ask price and high bidder are determined as follows:
    a) Compare the current high bidder's maximum bid and the new bid. The higher bid determines the new high bidder. A proxy bid is placed on behalf of the new high bidder which becomes the new ask price.
    b) The high bidder's proxy bid is typically the second highest bidder's maximum bid plus one bid increment.
    c) The exception to (b) occurs when the high bidder's maximum bid is less than one bid increment over the second highest bidder's maximum bid. In this case, the high bidder's proxy bid, and the ask price, is set to his maximum bid.
v. When two bids are for the same amount, the earlier one takes precedence.
vi. A bidder cannot lower his previous maximum bid.
vii. A bidder can raise his bid when he is winning, and in general, it will not change his proxy bid. The exception is when the bidder is winning and in condition (iv.c) above. In this case, if the winning bidder raises his offer, eBay would raise the current price to the minimum increment above the second highest maximum bid.

Table 1. A sequence of bids. The four columns on the right are computed from the data in the left columns. The actual value of the last excess increment is unknown, but a minimum can be computed.

Bid Number	Bid Time	Bidder ID	Bid Amount	Excess Increment	New High Bidder	New Ask Price	New Minimum Bid
1	June-17-01 15:06:20	63246	89.99	0	63246	89.99	90.99
2	June-17-01 15:53:20	59729	95	4.01	59729	90.99	91.99
3	June-17-01 17:51:12	59207	95	3.01	59729	95	96
4	June-17-01 17:53:06	59207	96	0	59207	96	97
5	June-17-01 17:56:57	45020	100	3	45020	97	98
6	June-17-01 17:58:32	59207	98	0	45020	99	100
7	June-17-01 17:58:45	59207	100	0	45020	100	101
8	June-17-01 18:01:08	45020	102.5	2.5*	45020	102.5	105

Table 1 shows how ask price and excess increment of bids are calculated for a sample auction. The data is from a real GBA auction, though the bidder IDs have been anonymized. The auction is a 3-day auction that started on June 14, 2001 with a scheduled end on June 17 at 18hr 3min and 14sec PST. The starting price set by the seller was $89.99.

[4] We found one exception to this rule. If the bid is submitted within a second of the previous bid, eBay may accept the later bid before processing the prior bid. Thus, we occasionally see bids that are lower than the new minimum.

Bidder 63246 places the first bid of $89.99. The minimum bid and ask price are equal at the start of the auction (by Step i). Hence, the excess increment for this bid is zero. After this bid is processed, the high bidder is 63246. The first bid does not change the ask price (by ii). However, the new minimum bid becomes $90.99. (in this price range, eBay requires a $1 increments; above $100 the increment rises to $2.50). Bidder 59729 places the second bid at $95 and becomes the high bidder (by iv.a). The excess increment is $4.01—the difference between the minimum bid ($90.99) and bid amount ($95)—and the new ask price is one bid increment over the second highest bid (by iv.b) i.e., $89.99 + $1.00. Bidder 59207 places the third bid and ties with 59729's hidden maximum bid. Because bidder 59729 placed his bid earlier, he remains the high bidder (v), and so bidder 59207 raises his bid to become the high bidder. Later in the auction, we see that when bidder 45020 places the eighth bid, he was already the current high bidder (by the tie breaker). Generally, this would not change the ask price. This case, however, is the exception to rule (vii) and eBay raises the current price to one full increment over $100 i.e., $102.5. The auction ends with bidder 45020 winning the item at $102.50. Note that we do not know the actual value of 45020's winning bid. The excess increment value for the winning bid (eighth bid) is a lower bound on the actual excess increment.

4 Analysis and Results

In addition to more straightforward views of the data (i.e., bid value charted over time), we found that examining graphs of excess increment values for individual bidders to be a useful way to visualize some aspects of a bidder's strategy. Figure 1 represents a graph of excess increment values of all bids submitted by bidder 62013 across all 3-day auctions in which he participated. The horizontal axis denotes the time of the bid from the auction's end (in seconds). The vertical axis denotes the excess increment value of the bids (in $). All bids for a specific auction are connected by a dotted line and each auction is represented with a different color. An 'h' above a marker denotes that the bidder became the high bidder after this bid was processed. Analyzing bidding behavior often required going back and forth between different views to construct a complete picture of the auction context.

We observe in Figure 1 that in several auctions bidder 62013 bid repeatedly in a short span of time, with increasing excess increment values, until he became the high bidder. This is one of several patterns that commonly appear in the data. To measure the actual frequency of these patterns in the restricted data, we developed tests to label individual engagements. Recall that an engagement is the set of all bids by an individual bidder in an individual auction. There are 49,523 engagements in D_r, varying in size from 1 to 24 bids. Let E denote the set of all engagements in D_r, and define $E_n \subseteq E$, to be the subset of engagements in which the bidder has placed exactly n bids in that particular auction. Figure 2 shows the distribution of engagement sizes. About 66% of engagements belong to subset E_1. The rest of the engagements exhibit multiple bids, with nearly 15% having three or more bids.

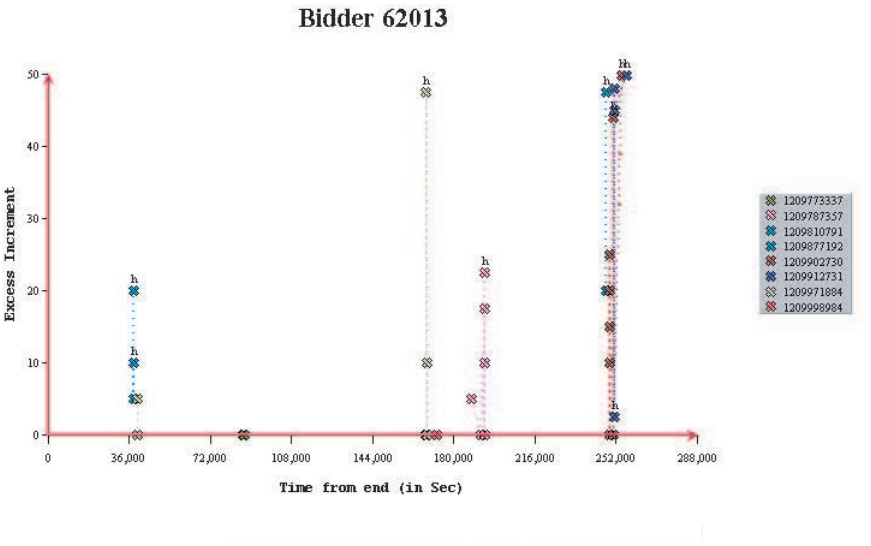


Fig. 1. Bidder 62013's bidding behavior in eight auctions.

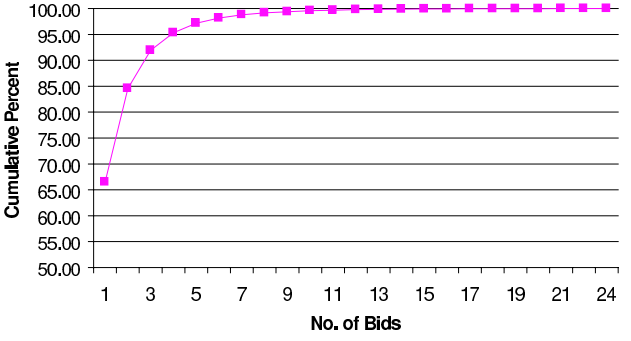


Fig. 2. Number of bids in engagements.

4.1 Single Bid Engagements (E_1)

We find that, irrespective of the auction duration, a large number of single-bid engagements have bids placed near the end of the auction, a behavior we refer to as *late bidding*. Nearly 58% of the bids were classified as late bidding. Figure 3 shows the distribution of bids during the last hour. A significant fraction of these bids are submitted in the closing seconds of the auction, a practice called *sniping*. This behavior arises despite advice from both auction theory and eBay itself that bidders should simply submit their maximum willingness to pay, once, early in the auction. Hence, it is inviting to attribute sniping behavior as being primarily due to naïve, inexperienced or irrational behavior. However, Roth and Ockenfels [2,3] show that the sniping and late bidding need not result from

either irrational behavior or common value properties of the objects being sold. They study a model of eBay's marketplace in which expert buyers have a collusive equilibrium in which they snipe. Although some bids are submitted too late to be accepted, the net effect is that prices are lowered enough by the decreased competition to compensate for the occasional rejected bid.

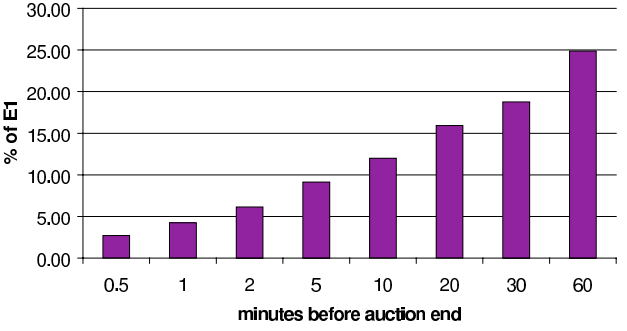


Fig. 3. Distribution of bids during the last hour.

Bidders who do seem to follow eBay's advice are also identifiable in E_1. We refer to the bidder classification by Bapna, et al. [4] in which they identify a bidder type called *Evaluators*. Evaluators are modeled as users who have a clear idea of their valuation and have the following characteristics:

- They bid once, at a high value, and well before the end of the auction.
- Their bids are usually significantly greater than the minimum required bid at that time.

To differentiate late bidders from evaluators, we restrict our attention to only those engagements in which the single bid was placed at least one day before the end of the auction (E_1^*). However there are several reasons that make identifying evaluators non-trivial. First, it is not uncommon that the game consoles auctioned on eBay are bundled with other things like games or extra joysticks that affect buyer's willingness to pay. Second, the bidder's valuation may change over time as the market price decreases.

To identify evaluator behavior, we group engagements in E_1^* according to bidder and type of item (i.e., PS2 or GBA). The standard deviation of the group of bids is calculated. We then consider two cases. First, if the standard deviation falls within a specified lower value, σ_{lower}, we classify the corresponding engagements as an evaluator's behavior. Figure 4 compares the bid values and the ask prices for bidder 45122's six engagements in E_1^*. The standard deviation of the bids in these six auctions is 0.52. We observe that the bids are significantly greater than, and apparently independent of, the ask price at the time.

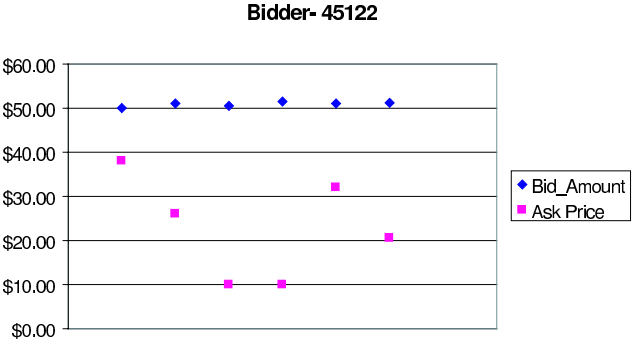


Fig. 4. Example of evaluator behavior in six auctions.

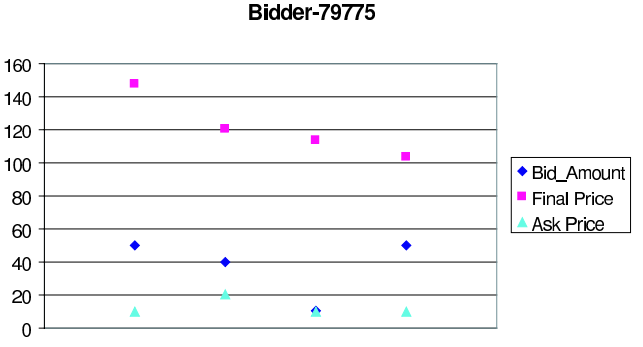


Fig. 5. Another example of evaluator behavior across four auctions.

The second case involves behaviors where the standard deviation is higher than σ_{lower} but less than an upper value, σ_{upper}, and is designed to capture fluctuations due to the content of the auctioned bundles. The test assumes that the final price of the auction reflects the true market value of the object. When $\sigma_{lower} < \sigma < \sigma_{upper}$, we examine the correlation of the bid amounts with the final price of the corresponding auctions (r_{final_price}) and with the ask prices of the auction at bid time (r_{ask_price}). If there is a strong correlation between bid amounts and final prices, we assume the deviation in bid amounts is attributable to different options in the sellers' offerings. Further, a weak correlation between bid amounts and ask prices support classifying the behavior as evaluator behavior because it suggests that the bidder's action does not depend on the ask price. Figure 5 shows an example having such characteristics. In this case, r_{final_price} = 0.30 (Strong-Weak association) and r_{ask_price} = 0.08 (Little, if any association). Thus, we also classify this case as depicting evaluator behavior.

Keeping σ_{lower} = 10.00, σ_{upper} = 20.00, $r_{final\_price} > 0.2$ and $r_{ask\_price} < 0.2$, we have 13% of E_1 (comprising of 1074 bidders; PS2 - 501 and GBA - 573), exhibiting the

evaluator behavior. There is a scope for improving this analysis by using better statistical methods. It would also be interesting to explore other explanations for such behavior.

4.2 Multiple Bid Engagements

Multiple-bid engagements, $E_{>1}$, account for more than 30% of E. One behavior common in $E_{>1}$ is the *skeptic* behavior. We define the skeptic as a bidder who submits multiple bids, all of which have zero excess increment. That is, the bidder always bids the minimum increment over the current ask price. One explanation for this behavior is that the bidder might be naïve or skeptical of eBay's proxy system and hence would always bid the minimum acceptable bid. Figure 6 shows a bidder exhibiting skeptic behavior. He participated in four auctions, submitting multiple bids all of which had zero excess increment. Around 18% of E_2 (1658 out of 8936) are classified as skeptics. In the remaining $E_{>2}$ group nearly 10% (738 out of 7649) follow the skeptic pattern.

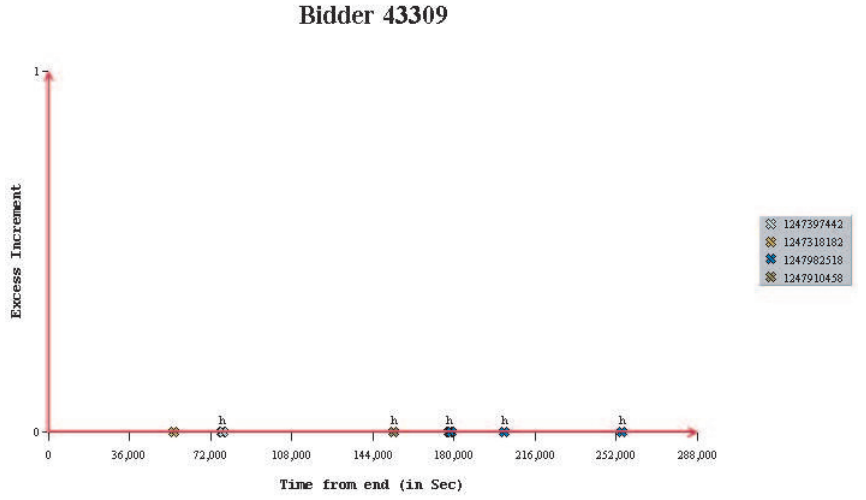


Fig. 6. Example of skeptic behavior.

Unmasking is another behavior that is common in $E_{>2}$. The pattern, shown in Figure 1, involves a series of closely placed bids, with variable excess increments, often terminating in a tentatively winning offer. There are several possible explanations for this behavior, including simply attempting to become the highest bidder. In the PS2 and GBA domains, there are many alternative auctions for the same item in which to bid. A bidder may use the unmasking technique to expose the maximum bid of someone else's proxy bid or until he has enough information to decide to move on to another auction.

Finally, bid patterns like unmasking are suggested by eBay to possibly indicate shill bidding (described in Section 4.3).[5]

We identify unmasking behavior by the following characteristics:

- The bidder places multiple bids in a short span of time. (We vary the time span from 2 min to 10 min)
- There are no bids by other bidders in-between these bids.
- At least one bid in the engagement has an excess increment greater than zero.
- At least two non-winning bids are submitted as part of the sequence.

We find the unmasking behavior in 40%-43% of the $E_{>2}$, when we vary the time of successive bids from 2 minutes to 10 minutes. The fact that the vast majority of this behavior is identifiable with a 2-minute window suggests that it is a standard tool in the bidders' bag of tricks.

Around 5% of $E_{>2}$ engagements have behavior in which the excess increment of successive bids decreases. Figure 7 is representative of engagements in this class, showing a typical behavior that appears as slanted lines. This behavior is probably attributable to a bidder who periodically attends to the auction to check whether he is outbid. The excess increment decreases as the end of the auction approaches because the price is increasing. Although this type of behavior was easy to identify by visual inspection, we have not yet identified the key attributes that form a coherent cluster for automatic detection.

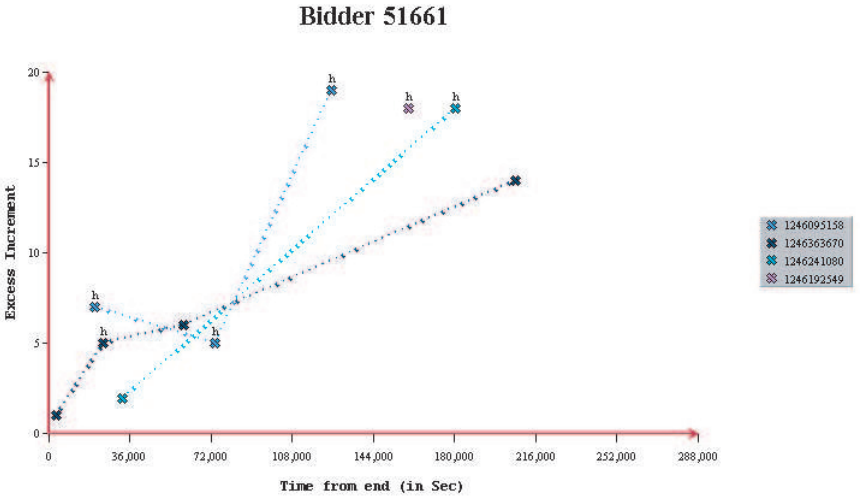


Fig. 7. Engagements of Bidder 51661.

[5] See eBay's help center under the topic "shill".

4.3 Identifying Types of Bidders

The above classifications were derived largely by hand and somewhat blurred the line between classifying engagements and classifying bidders. In this section, we re-establish the distinction and investigate whether we can identify types of bidders based upon their engagements using a clustering algorithm. Using variations of the above classification rules,[6] we labeled each engagement as either an evaluator, a skeptic, a snipe, an unmasking, or as unidentified. We then constructed a *bidder profile* for each bidder which we represented as a vector where the elements of the vector were the proportion of engagements of each type. For instance, if buyer A had two engagements that classified as skeptic, one unmasking, and one unidentified, her vector would be $\langle 0, 0.5, 0, 0.25, 0.25 \rangle$.

Using the SAS Enterprise Miner™ software, we ran a standard cluster analysis on the bidder profiles. Table 2 shows the results when we set the desired number of clusters to five. The first cluster consists primarily of bidders whose engagements did not fall into one of our categories. This cluster represents 88% of the bidders. However, the other 12% of the bidders do cluster nicely into the four groups with centroids firmly in one type of bidding behavior.

Table 2. Results from clustering the bidder profiles.

Cluster	Frequency	Evaluate	Skeptic	Snipe	Unmask	Unidentified
1	18939	0.0037	0.0001	0.0193	0.0150	0.9619
2	227	0.9793	0	0.0103	0.0066	0.0038
3	1377	0.0002	0	0.9711	0.0025	0.0263
4	798	0	0	0.0008	0.9772	0.0220
5	12	0	1	0	0	0

We draw a couple of conclusions from this analysis. First, consistent skeptic behavior was extremely uncommon, perhaps because it is unlikely to be successful or the bidding population is simply more sophisticated. Second, perhaps not surprisingly, snipers were the most common identifiable cluster. However, it is surprising that the cluster representing bidders who consistently use unmasking as a strategy was quite large—more than half as big as snipers. Finally, evaluators make up only about 1% of the sampled bidding population.

When we reduce the number of desired clusters, Enterprise Miner drops clusters from the list in decreasing order. When we increase the number of desired clusters to, for example, seven, about 1300 bidders move from cluster one into hybrid groups in which about half of the engagements are unidentified, and the other half fit in one of the other categories.

[6] Variations of the rules were used when the rules required looking at multiple engagements. In particular, we approximated the evaluator rule by assuming an engagement that consisted of a single bid that occurred at least a day before the auction's end and which was greater than 90% of the clearing price was an evaluator bid. We also extended the skeptic category to engagements with only one bid.

5 Fraud Detection

Incidents of fraud in online auctions are increasing rapidly; according to the Internet Fraud Complaint Center, online auction fraud is the number one type of Internet fraud [5] and accounts for 46% of all complaints. One of the more subtle types of auction fraud to detect is *shilling*. Shilling, also known as colluding or bid rigging, is a method by which sellers try to hike up the prices in an auction by placing buy bids under aliases or through associates. Shilling is often very difficult to detect on public markets primarily because of the sheer number of online auctions and users, and the ease with which users are able to create new accounts. Experienced shills can easily disguise themselves among this overwhelming amount of data.

The following attributes are claimed to be indicators of shilling behavior, and are particularly applicable to our eBay data:[7]

1. There is a strong association between a seller and a buyer, or a ring of buyers. By association, we mean that the shill(s) appear very frequently in auctions hosted by the seller.
2. The shill wins the auctions infrequently, if at all. An aggressive shill may occasionally win the auction, but since he is working for the seller, the two parties need never consummate the deal.
3. The shill would eschew sniping and very late bidding as this would put him at risk of winning the auction and not permit the real buyers enough time to respond to his bids.
4. A shill is likely to bid repeatedly to increase the price in steps. In our data, this is likely to appear as the unmasking strategy.

To test the feasibility of identifying shilling, we performed association analysis between the users (sellers and buyers) participating in the auctions. We construct an abstraction of the engagement data with an entry for each buyer-auction and seller-auction pair. The user-ids of the sellers were prefixed with the letter 's' while those of the buyers remained the same. For example, the entry {465768718, s71497} indicates that seller 71497 participated in auction 465768718, while the entry {465768718, 59493} indicates that buyer 59493 participated in the same auction.

We used SAS Enterprise Miner for the association analysis. A rule like A $\Rightarrow$ B suggests that the presence of user A in an auction predicts the presence of user B. The *support* is the number of times A and B appeared together in an auction divided by the total number of auctions (approximately 7000 in D_r). In our data set, the support is quite low and not particularly instructive because of the number of auctions. The *confidence* is the number of times A and B appeared together in an auction divided by the number of times A appeared in an auction. Finally, the number in the *transactions* column is the number of auctions in which A & B both participated. With a minimum confidence of 75% and a minimum of five transactions, 79 two-item rules were found representing

[7] Descriptions of common features of shilling behavior are available on sites like AuctionWatch (http://www.auctionwatch.com).

Table 3. Sample binary rules generated by association analysis.

Rule	Confidence	Transactions	# won
43645 ⇒ s60993	100	26	1
s60993 ⇒ 43645	96.3	26	1
s43998 ⇒ 62466	76.92	10	0
62466 ⇒ s43998	83.33	10	0
77812 ⇒ s43998	80	8	0
49153 ⇒ s43998	100	6	0
59949 ⇒ s43998	100	5	0
56146 ⇒ s49449	100	5	0
45581 ⇒ s50336	86.21	25	2
66240 ⇒ s50336	92.86	13	1
70053 ⇒ s50336	91.67	11	1
68111 ⇒ s50336	83.33	10	0
56992 ⇒ s50336	90	9	1
68746 ⇒ s50336	100	6	0
82377 ⇒ s50336	100	5	0
42628 ⇒ s50336	100	5	2
64621 ⇒ s50336	100	5	0
67124 ⇒ s50336	100	5	1
71139 ⇒ s50336	100	5	1
47319 ⇒ s50336	83.33	5	1
55477 ⇒ s53124	100	5	3
64512 ⇒ s53666	80	12	0
50732 ⇒ s53666	100	11	0
64346 ⇒ s53666	100	10	0
77145 ⇒ s53666	85.71	6	0

69 different buyers. Table 3 shows a subset of the results, clustered by seller ID. The rightmost column indicates the number of times the buyer was the high bidder among the transactions. Ten buyers appear in two rules because the complementary relationships (B ⇒ A) was also found to have high confidence. No buyer had a strong relationship with more than one seller.

It is clear from this data that there were strong relationships between some sellers and buyers. For example, there is high confidence for the two rules that involve seller s60993 and buyer 43645. All 26 auctions in which buyer 43645 bid were hosted by s60993, and all but one of seller s60993's auctions had buyer 43645 participating. Furthermore, 43645 won only once in 26 tries. This result certainly suggests that s60993 may be using 43645 as a shill, but that is not the only explanation. For example, s60993 may be a well established commercial seller, or may be physically located near a particular buyer. Further research would be necessary by eBay or by a law enforcement agency to determine whether these results pinpoint fraudulaent behavior. Other suspicious cases seem to involve more sophisticated use of shills. Seller s50336 had strong relationships with 12 different bidders and sellers s43998 and s53666 each had four strongly related bidders, none of whom won any auctions. Table 4 shows some of the 3-way rules found

Table 4. Sample 3-way rules generated by association analysis,

Rule	Confidence	Transactions
77889 ⇒ s60993 & 43645	54.5	6
s60993 & 77889 ⇒ 43645	100	6
77889 & 43645 ⇒ s60993	100	6
77889 ⇒ s60993 & 43645	54.5	6

by Enterprise Miner. Results like these may suggest that seller s60993 is using two shills in the same auction, or 43645 is the more likely shill, having greater confidence and support. Thus, the existence of the 3-way relationship may actually lower our belief that 77889 is a shill. No such strong 3-way rules were found for seller s50336, which suggests that the 12 buyers bid is largely disjoint sets of s50336's auctions—a behavior we would expect to see if they indeed were shills.

This analysis incorporated only the simplest attributes of the relationship between buyers and sellers. Including other attributes may enable more precise classifications. For example, our analysis did not consider the time at which the auctions were held. We expect that a relationship that remains stable over a long period of time would be more suspicious. In addition, our data set was restricted to two auction categories only (GameBoy and PlayStation 2); a seller may potentially be involved with several categories and data about a seller-bidder association over these categories could be more evidence. To rule out the other plausible explanations, we could investigate the seller's eBay page for features, such as geographical proximity to the seller or commercial reputation, that might make him a favorite of the seller. An examination of the feedback written by the buyer and seller may also be useful evidence, particularly negative feedback serving as an anti-correlant with shilling. Finally, information about whether a transaction between a seller and a buyer actually occurred is not available to us. Again, we would expect that the consummation of a transaction would be negatively correlated with shilling. In the case of these particular items, it would also be suspicious to see a buyer to continue to bid in auctions even after winning an object, though such behavior could also be explained by the first seller's failure to deliver the object.

Most significantly, to pursue this line of research further requires not only the bid data, but also the results of real investigations to determine which relationships truly represent fraud, and which ones have other explanations. It is rumored that eBay does police its sellers, but the techniques they use and the results of their investigations are not made public.

5.1 The Relationship between Shills and Bidding Behaviors

One of the claimed indications of shill bidding was behavior consistent with what we have identified as unmasking. To determine if unmasking, or any other strategy, was an indicator of the presence of a suspected shill, we examined the bidding behavior of those buyers identified as shills. The 69 bidders identified above participated in a total of 749 engagements. Table 5 shows the frequency of each type of bidding strategy among the 749 engagements. As expected, sniping is not a common behavior among these potential

Table 5. Frequency of bidding strategies among the suspicious bidders.

Behavior	Frequency
Evaluate	12
Skeptic	0
Snipe	5
Unmask	32
Unidentified	700

shillers and it occurs much less frequently than in the larger population. However, none of the other bidding behaviors seem particularly predictive of the presence of a shill, and unmasking, in particular, occurs a little less frequently among the alleged shills than it does among the general population (about 4% versus 6%).

6 Related Work

Internet auctions have attracted the attention of auction theorists and econometricians, and several recent papers have examined online markets. Ebay, being the most popular site, is the first choice of many such studies. David Lucking-Reiley, et al. [6,7] analyze the effect of various eBay features on the final price of auctions. They find that a seller's feedback rating has a measurable effect on the final price, with negative comments having a much greater effect than positive comments. Houser and Wooders [8] find a similar effect of the feedback ratings on the auction price. Roth and Ockenfels [2] observe late bidding in online auctions and suggest that multiple causes contribute to late bidding, with strategic issues being related to the rules about auction ending. Further, they show that late bidding may be an equilibrium for both public and private valued goods, though the explanation is stronger in situations in which "experts" find it beneficial to snipe rather than to bid early and thereby lend their appraisal to the authenticity or value of the object. Ünver [9] analyses the evolution of strategic multiple and last minute bidding using artificial agents. The work found support for multiple bidding in both private-value and common-value models. Bapna, et al. [4] reveal that the traditional theoretical assumptions regarding the homogeneity of bidder strategies are violated in online auctions. This conclusion is supported by our analysis as well.

Researchers have recently begun studying both the social aspects of auction fraud, and mechanisms to combat it. Chua and Wareham [10] describe some of the actions being taken by the auction community to combat fraud. Among other things, auction communities have established formal institutions, such as the Online Auction Users Association, that lend credibility to their members. In addition, vigilantes have taken it upon themselves to identify and sometimes even punishing suspected defrauders. In another paper, Chua, et al. [11] propose primarily administrative mechanisms to help address fraud in Internet auctions. Wang, et al. [12] take an incentive engineering approach to the problem, and propose a mechanism in which the auction house collects a commission from the sellers that is computed in such a way that it makes shill bidding unprofitable for the sellers. Of course, the fee itself is a form of friction, and reduces the efficiency of the overall market.

7 Conclusion and Future Work

This paper serves as an exploratory analysis of the feasibility of various data mining tasks on data collected from eBay. We introduce the concept of excess increment; a useful attribute of bids that is indicative of bidding strategies. We are able to identify sniping, late bidding, evaluator, skeptic and unmasking behaviors. The former three strategies have been previously suggested by other researchers, and the latter two strategies are newly identified. These behaviors account for a significant portion of engagements in our sample data and confirm that varieties of bidding strategies are common on eBay. Unmasking, in particular, is a newly identified strategy that may have an interesting economic explanation.

Our longer term goal is to develop or apply clustering techniques that are able to automatically group bidders into the above, or other, categories. This paper takes an empirical approach to identify bidding strategies and shows that data mining techniques may be used to enhance the results. It is also important to note that a variety of attributes must be examined to automatically classify bidders' behavior. Some strategies, like unmasking, exist within a single engagement, while others, like evaluators, require examining a bidder's behavior across multiple engagements. However, we show that with a relatively coarse method of classifying engagements, a clustering algorithm can successfully identify groups of bidders who behave in a consistent manner.

Shill detection in online auctions is an interesting and important extension to this work. Currently, any actions that auction houses like eBay take to detect or punish shill behavior are opaque to the user community. The techniques we develop in this paper could easily be integrated into a tool and made available to the user community. However, such a tool would engender other problems, including angering legitimate sellers who are incorrectly labeled as suspects, and providing information that enables swindlers to become more sophisticated.

Finally, we plan to use the data collected in this study, and the models of bidding strategies, to improve the realism of simulations of economic markets. These simulations might help us to develop software agents that facilitate effective participation in online markets.

Acknowledgements. This work was supported by the National Science Foundation under CAREER award No. 0092591 and the e-Commerce@NCSU program. In addition, we would like to thank Jon Doyle, for his role as committee member and data mining instructor, and the members of the Intelligent Commerce Research Group at NCSU for their encouragement and critiques.

References

1. Shah, H.S., Joshi, N.R., Wurman, P.R.: Mining for bidding strategies on eBay. In: WEBKDD 2002 Workshop, Edmonton, Alberta, Canada (2002)
2. Roth, A.E., Ockenfels, A.: Last-minute bidding and the rules for ending second-price auctions: Evidence from eBay and Amazon auctions on the Internet. American Economic Review **92** (2002) 1093–1103

3. Ockenfels, A., Roth, A.E.: The timing of bids in internet auctions: Market design, bidder behavior, and artificial agents. AI Magazine **23** (2002) 79–87
4. Bapna, R., Goes, P., Gupta, A.: A theoretical and empirical investigation of multi-item on-line auctions. Information Technology and Management **1** (2000) 1–23
5. Internet Fraud Complaint Center: 2002 internet fraud report. Technical report (2002)
6. Lucking-Reiley, D., Bryan, D., Prasad, N., Reeves, D.: Pennies from eBay: The determinants of price in online auctions. Technical report, University of Arizona (2000)
7. Katkar, R., Lucking-Reiley, D.: Public versus secret reserve prices in eBay auctions: Results from a pokmon field experiment. Technical report, University of Arizona (2000)
8. Houser, D., Wooders, J.: Reputation in auctions: Theory and evidence from eBay. Technical report, University of Arizona (2001)
9. Ünver, M.U.: Internet auctions with artificial adaptive agents: Evolution of late and multiple bidding. Technical report, Koc University, Istanbul, Turkey (2001)
10. Chua, C., Wareham, J.: The covert war on internet auction fraud. Technical report, Georgia State University (2002)
11. Chua, C.E.H., Wareham, J., Robey, D.: Anti-fraud mechanisms in internet auctions: The roles of markets, hierarchies and communities of practice. Technical report, Georgia State University (2002)
12. Wang, W., Hidvegi, Z., Whinston, A.B.: Shill bidding in english auctions. Technical report, Emory University (2001)

Automatic Categorization of Web Pages and User Clustering with Mixtures of Hidden Markov Models

Alexander Ypma and Tom Heskes

SNN, University of Nijmegen
Geert Grooteplein 21,
6525 EZ Nijmegen, The Netherlands
{ypma,tom}@mbfys.kun.nl
www.mbfys.kun.nl/~ypma

Abstract. We propose mixtures of hidden Markov models for modelling clickstreams of web surfers. Hence, the page categorization is learned from the data without the need for a (possibly cumbersome) manual categorization. We provide an EM algorithm for training a mixture of HMMs and show that additional static user data can be incorporated easily to possibly enhance the labelling of users. Furthermore, we use prior knowledge to enhance generalization and avoid numerical problems. We use parameter tying to decrease the danger of overfitting and to reduce computational overhead. We put a flat prior on the parameters to deal with the problem that certain transitions between page categories occur very seldom or not at all, in order to ensure that a nonzero transition probability between these categories nonetheless remains. In applications to artificial data and real-world web logs we demonstrate the usefulness of our approach. We train a mixture of HMMs on artificial navigation patterns, and show that the correct model is being learned. Moreover, we show that the use of static 'satellite data' may enhance the labeling of shorter navigation patterns. When applying a mixture of HMMs to real-world web logs from a large Dutch commercial web site, we demonstrate that sensible page categorizations are being learned.

Keywords: Web usage mining, data mining, navigation patterns, automatic categorization, clustering, hidden Markov models, mixture models, user profiling

1 Introduction

Each visitor of a web site leaves a trace in a log file of the pages that he or she visited. Analysis of these click patterns can provide the maintainer of the site with information on how to streamline the site (connecting 'remote' pages that are often visited cooperatively, detecting common 'exit traces'), or how to personalize it with respect to a particular visitor type. However, due to the massive amount of data that is generated on large and frequently visited web sites, clickstream

O.R. Zaïane et al. (Eds.): WEBKDD 2002, LNAI 2703, pp. 35–49, 2003.

analysis is hard to perform 'by hand'. Several attempts have been made to learn the click behaviour of a web surfer, most notably by probabilistic clustering of individuals with mixtures of Markov chains [1,12,13] Here, the availability of a prior categorization of web pages was assumed; clickstreams are modelled by a transition matrix between page categories. However, manual categorization can be cumbersome for large web sites. Moreover, a crisp assignment of each page to one particular category may not always be feasible.

In this paper we extend this existing approach by learning the most likely categorization of a page along with the inter-category transitions, i.e. we model a clickstream of a particular surfer type by a hidden Markov model (HMM). In order to incorporate heterogeneity of users (there will be several types of surfers on a web site), we postulate a *mixture* of HMMs (mHMM) to model clickstreams. In our formulation we make the membership of a user to a user type explicit, which then allows for inclusion of additional user data.

In the following section we introduce the problem with a small example, and then describe the model for clustering of web surfers. We give the update equations for training the mHMM model with the Expectation-Maximization algorithm and describe how to incorporate prior knowledge and additional (static) user information. Then we apply the method to artificial data and to logs from a large commercial Dutch web site, discuss both method and results and draw conclusions.

2 Mixtures of Hidden Markov Models for Web Usage Mining

In order to clarify the web mining problem, we start with a small example.

2.1 Web Mining: An Example

Consider a web log file, that contains entries of the following form:

```
194.79.42.11 - [06:54:49] "GET /common/graphics/tools_catalog_on.gif HTTP/1.0" 304 0 "-"
146.8.233.251 - [06:55:03] "HEAD / HTTP/1.0" 500 305 "-"
217.142.71.136 - [06:55:02] "GET /br1/custsvc/cs_category_list.jsp
209.199.168.175 - [06:54:50] "GET /en/financiele/results.html HTTP/1.0" 404 207 "-"
146.8.233.251 - [06:54:58] "HEAD / HTTP/1.0" 302 0 "-"
213.79.178.45 - [06:54:42] "GET /common/graphics/products/product.gif HTTP/1.0" 200 1842 "-"
194.79.42.11 - [06:54:49] "GET /common/graphics/welcome HTTP/1.0" 304 0 "-"
146.8.233.251 - [06:55:04] "HEAD / HTTP/1.0" 302 0 "-"
217.142.71.136 - [06:54:40] "GET /common/tools_header.gif HTTP/1.0" 200 353 "/br1/index.jsp"
```

We want to derive information about the click behaviour of web users automatically from this file[1], since manual processing is not doable. We assume that each web user can be uniquely identified; in this paper we make the (simplifying) assumption that a user's IP-adress acts as a unique identifier. Each different page that is requested from a web site gets a unique page id (which is only based on

[1] The IP-adresses and URL-s in the displayed excerpt were anonymized, so any trace to actual persons or web-sites is coincidental

its URL, not on additional page information like keywords or a semantic description). If we retain the ordering in the page requests and if we assume that a time difference of 30 minutes between two http-requests of the same user indicates different sessions, we end up with several (possibly intertwined) clickstreams of the following form:

```
Y11 = 8 9 10 11 11 11          Y12 = 8 8 9 12 13 13 14 15 14
Y21 = 1 2 4 3 5 7 6 5 7 6 4 3  Y22 = 1 2 2 2 2 4 2 4 3
```

where $\mathbf{Y^{1,j}}$ and $\mathbf{Y^{2,j}}$ are clickstreams no. $j = 1, 2$ for certain **user 1** and **user 2**, respectively. The problem is now to assign a new clickstream, like

```
Ynew = 8 9 9 10 11 11 153 154 155 9 9 9 8 9 10 11 11 11 11 24
```

to the user type that it resembles best. We learn a model for each user type k from a set of training patterns $\{Y_t^{i,j}\}$ and we assume that click behaviour can be described by modelling transitions between **page categories** $X_t \in 1, \ldots, m$, rather than between individual pages $Y_t^{i,j} \in 1, \ldots, M$. This gives a computational advantage since there are less categories than individual pages ($m < M$) and it is more meaningful to model inter-category transitions than inter-page transitions. To continue our example, one type of user (e.g. "general-interest user") may typically start by doing a *search*, followed by *downloading* a report or a piece of software and finally checks the *latest corporate news*. Another type of user (e.g. "device-interest user") may first enter the pages concerning the *retail shop* of the company, then he browses through the set of *consumer products*, tries to *troubleshoot* the device that he owns and concludes by checking out new *promotions* offered by the company. Here, the *italic* phrases are page categories and a particular web page will be assigned to one (or a few) categories.

In our approach, we learn the probability that a user of type k jumps from page category i to category j, which is represented by entries in the transition matrix $A^k(i, j)$. Moreover, we learn the categorization of a web page by determining the probability that a certain page $Y_t = l$ is observed in a clickstream at time t, given that the page category at time t is $X_t = i$. This is represented by entries in the observation matrix $B^k(i, l)$; if we assume a common categorization for all user types, we end up with just one common observation matrix B.

2.2 Problem Formulation

Consider a population of web surfers, denoted by $\{i\}, i = 1, \ldots, N$. Each surfer i generates n_i clickstreams $Y^{i,j} = \{Y_t^{i,j}\}, t = 1, \ldots, T_{ij}$, i.e. possibly unequal-length ordered sequences of pages that have been visited in a surfing-session. An element $Y_t^{i,j}$ denotes the observed page id (in the range 1, ... , M) at time t of clickstream no. j that is generated by user no. i. All clickstreams generated by surfer i are collected into $Y^i = \{Y^{i,1}, \ldots, Y^{i,n_i}\}$.

We assume K clusters of web surfers and our overall model Θ consists of separate models for each cluster, $\Theta = \{\Theta_k\}, k = 1, \ldots, K$. The cluster label ("surfer type") is given by the variable $C \in 1, \ldots, K$. We model the dynamics

in the traces at the level of page categories X, which are hidden state variables (unknown beforehand) in a hidden Markov model. The inter-category transitions are governed by a (cluster dependent) transition matrix A^k and the initial state distribution vector Π^k, the categorization of a page is given by the observation matrix B^k. If we have additional (static) information about a surfer (like demographic information, area code, etc.) this may also give us an indication about the surfer type. If the B matrix is *shared* (section 2.5), we have the graphical model shown in figure 1.

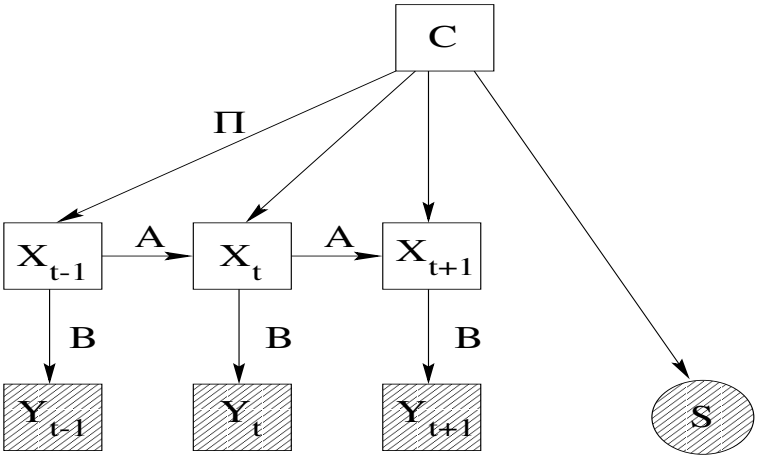


Fig. 1. A mixture of hidden Markov models and additional user data, presented as a graphical model. The variables Y_t (discrete) and S (continuous or discrete) are observed, the category variables X_t (discrete) are hidden. Note that in this figure the observation matrix B is independent of the cluster label C (which occurs with a shared B matrix), since there are no arrows from C to the Y-nodes. Furthermore, note that S and Y are independent given C.

2.3 Dynamic Model

We now address the dynamic part of the model (with cluster-dependent observation matrices B^k), the static part and the sharing of B is included in section 2.5. The likelihood of the (dynamic) data from all users is

$$P(Y|\Theta) = \prod_{i=1}^{N} P(Y^i|\Theta) \tag{1}$$

which expresses the assumption that users are independent of each other. The likelihood of the (dynamic) data of user i is given by the mixture

$$P(Y^i|\Theta) = \sum_{k=1}^{K} P(Y^i|c_i = k, \Theta_k)\alpha_k \tag{2}$$

where

$$P(Y^i|c_i = k, \Theta_k) = \prod_{j=1}^{n_i} P(Y^{i,j}|c_i = k, \Theta_k) \tag{3}$$

is the likelihood of user data i given that the user type is known. The latter equation expresses that different surfing sessions of a user are modelled as independent events given the user type[2]. The mixture coefficients obey $\sum_k \alpha_k = 1$. The likelihood of a particular sequence $Y^{i,j}$, given that user i is in cluster k, is given by the well-known quantity for HMMs

$$P(Y^{i,j}|c_i = k, \Theta_k) = \sum_X P(Y^{i,j}, X|c_i = k, \Theta_k) \tag{4}$$

where

$$\begin{gathered} P(Y^{i,j}, X|c_i = k, \Theta_k) = \\ \Pi^k(X_1)B^k(Y_1^{i,j}|X_1) \prod_{t=1}^{T_{ij}-1} A^k(X_{t+1}|X_t)B^k(Y_{t+1}^{i,j}|X_{t+1}) \end{gathered} \tag{5}$$

2.4 EM Algorithm

We can train (the dynamic part of) the model from the previous section using the EM algorithm. In the update equations we use the following definitions [8]:

$$\begin{aligned} \gamma_t^{i,j,k}(x) &= P(X_t = x|Y^{i,j}, \Theta_k) \\ \xi_t^{i,j,k}(x, x') &= P(X_t = x, X_{t+1} = x'|Y^{i,j}, \Theta_k) \end{aligned} \tag{6}$$

E-step. This Involves an Update of the (Hidden) Memberships c_{ik}.

$$c_{ik} := P(c_i = k|Y^i, \Theta) = \frac{\alpha_k P(Y^i|c_i = k, \Theta_k)}{\sum_l \alpha_l P(Y^i|c_i = l, \Theta_l)} \tag{7}$$

[2] Session-to-session effects are not always entirely explained by the user label. If significant user-specific session correlations are present, equation (3) should be adapted.

M-step. This involves an update of the parameters $\alpha_k, \mathit{\Pi}^k, A^k, B^k$:

$$\begin{aligned}
\hat{\alpha}_k &= \frac{1}{N}\sum_{i=1}^{N} P(c_i = k|Y^i, \Theta) \\
\hat{\mathit{\Pi}}^k(x) &= \frac{\sum_i c_{ik} \sum_{j=1}^{n_i} \gamma_1^{i,j,k}(x)}{\sum_i c_{ik} \sum_{j=1}^{n_i} \sum_x \gamma_1^{i,j,k}(x)} \\
\hat{A}^k(x,x') &= \frac{\sum_i c_{ik} \sum_{j=1}^{n_i} \sum_{t=1}^{T_{ij}-1} \xi_t^{i,j,k}(x,x')}{\sum_i c_{ik} \sum_{j=1}^{n_i} \sum_{t=1}^{T_{ij}-1} \gamma_t^{i,j,k}(x)} \\
\hat{B}^k(x,y) &= \frac{\sum_i c_{ik} \sum_{j=1}^{n_i} \sum_{t=1 \wedge Y_t^{i,j}=y}^{T_{ij}} \gamma_t^{i,j,k}(x)}{\sum_i c_{ik} \sum_{j=1}^{n_i} \sum_{t=1}^{T_{ij}} \gamma_t^{i,j,k}(x)}
\end{aligned} \tag{8}$$

2.5 Including Static User Data and Prior Information

As pointed out by Smyth [13], once the memberships are made explicit in a mixture model involving both dynamic data Y and static user data S, we can easily combine the two types of information to enhance the labelling of a user. If we assume that the dynamic and static data are independent given the cluster label, we may extend our original 'dynamic' mixture model (2) with static data to $P(Y^i, S^i|\Theta) = \sum_k P_k(Y^i, S^i|c_k, \Theta^k)\alpha_k$, leading to a modified E-step

$$c'_{ik} := P(c_i = k|Y^i, S^i, \Theta) = \frac{\alpha_k P_k(Y^i) P_k(S^i)}{\sum_l \alpha_l P_l(Y^i) P_l(S^i)} \tag{9}$$

The M-step equations for the dynamic and the static model separately remain the same, except that now the joint membership c'_{ik} is employed. It is very likely that additional information is not available for all surfers. In this case, we can set the probability of static data in cluster k to 1.

We remark that prior knowledge on the dynamics can be taken into account in the following manner [9]. Consider a reestimated transition probability of the form $\hat{P}(i,j) = n_{ij}/n_i$, with n_{ij} the transition count from state i to j and n_i the number of transitions from i. If our prior knowledge takes the form of an additional pseudo-sequence of length $\beta + 1$, which is divided into β_{ij} transitions from i to j, the Bayesian MAP estimate is

$$\hat{P}_{\mathrm{MAP}}(i,j) = \frac{n_{ij} + \kappa\beta_{ij}}{n_i + \kappa\beta_i}, \tag{10}$$

where $\beta_i = \sum_j \beta_{ij}, n_i = \sum_j n_{ij}$ and $0 \leq \kappa \leq 1$ determines the extent to which the prior or the data is used. A similar trick can be applied to the 'prior probability' $\mathit{\Pi}$ over states and the observation probabilities. Especially the latter quantity may easily tend to zero in cases with small sample sizes (limited number of observations, large dimensionality of the observables).

From our update formula for $\hat{B}^k$ it is clear that each mixture component has a private observation matrix. However, for our application the 'interpretation' of a category should preferably not be too different for different user types (e.g. 'Mercedes-Benz.html' should be categorized as 'cars', regardless the user's interests). Therefore, we constrain the observation matrices in all clusters to be equal (a.k.a. *parameter tying*). This has the additional advantage that we decrease the danger of overfitting in cases with a large number of observables (i.e. sites with many pages and relatively small number of visitors).

2.6 Computational Complexity

Since the datasets encountered in practice may be very large (a one-day log file of the web site analysed in section 3 is already .25 GB) the scalability of the proposed algorithm is an important issue. We mention that the computational complexity at the level of one EM iteration is

- linear in the total number of sequences $\sum_i n_i$ (and hence in the number of surfers N, assuming that the number of clickstreams per user can be bounded to a reasonable number),
- linear in the number of surfer types K,
- quadratic in the dimensionality of the hidden state space $|X|$, and
- linear in the number of samples in an individual clickstream T_{ij}

During each iteration, the costly step is the inference step where for each of the sequences (i.e. pairs (i, j)) and for each of the mixture components k the quantities $\gamma_t^{i,j,k}(x), \xi_t^{i,j,k}(x, x')$ and c_{ik} (for all surfers i) have to be computed. The cost of inference in an HMM for a particular component and sequence is $\mathcal{O}(|X|^2 T_{ij})$, see [5]. Hence, we see that the algorithm scales linearly with all relevant quantities except for the number of categories. However, we expect that for a certain web site the number of relevant categories will not depend too strongly on the number of considered clickstreams, i.e. if we consider twice as many clickstreams the appropriate number of categories will not increase as much (provided that the initial set of clickstreams was 'rich enough')[3].

By sharing the observation matrix we diminish the danger of overfitting. It is hard to predict how the likelihood surface changes when we include more data and larger models (and therefore how many iterations are necessary for convergence), though for mixtures of Markov chains this does not lead to superlinear behaviour [2].

[3] As an aside, a reviewer remarked that this assumption is indeed a plausible one. Furthermore, in this reviewer's experience, the categorization of pages is straightforward, even with dynamic page creation. This strengthens our assumption that meaningful categories may be obtained by automating the laborious task of page categorization, while at the same time allowing for a better scalable user clustering procedure

3 Experiments

3.1 Artificial Data

We generated artificial clickstreams using a 4-component mHMM. Every single HMM in the mixture model had 2 hidden states and was able to emit 3 observables. The A^k, B^k and Π^k parameters were markedly different for each component k. All 4 components were almost equally likely to occur. The generated sequences were distributed over 15 users, where each user received only sequences with the same label (so that the *user* could be labelled meaningfully as well). The resulting user labelling was: $\{1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 4, 4, 4, 4\}$. Users with label 1 produced 3 sequences, others produced 4 sequences. Each sequence had length 250 samples. We then trained an (oversized) 8-component mHMM

Table 1. Learned memberships from 4-component mixture data. Each entry in the table gives the membership c_{ik} of user i to component k; nonzero entries are in boldface.

$i \backslash k$	comp. 1	comp. 2	comp. 3	comp. 4	comp. 5	comp. 6	comp. 7	comp. 8
1	0	0	0.0	0	0.0	**0.0049**	0	**0.9951**
2	0	0	0.0	0	0.0	**0.0003**	0	**0.9997**
3	0	0	0.0	0	0.0	**0.0001**	0	**0.9999**
4	0	0	**1.0**	0	0.0	0.0000	0	0.0000
5	0	0	**1.0**	0	0.0	0.0000	0	0.0000
6	0	0	**1.0**	0	0.0	0.0000	0	0.0000
7	0	0	**1.0**	0	0.0	0.0000	0	0.0000
8	0	0	**1.0**	0	0.0	0.0000	0	0.0000
9	0	0	**1.0**	0	0.0	0.0000	0	0.0000
10	0	0	0.0	0	**1.0**	0.0000	0	0.0000
11	0	0	0.0	0	**1.0**	0.0000	0	0.0000
12	0	0	0.0	0	0.0	**0.9998**	0	**0.0002**
13	0	0	0.0	0	0.0	**1.0000**	0	0.0000
14	0	0	0.0	0	0.0	**0.9986**	0	**0.0014**
15	0	0	0.0	0	0.0	**1.0000**	0	0.0000

on these sequences for 1000 cycles in order to check whether our model was able to learn the correct parameters and user labelling, even if the number of components present in the data was smaller than the number of components in the model. We observed the learned memberships shown in table 1. It is clear that our model learns the correct user labelling; moreover, the number of underlying mixture components can be estimated from this table as well.

In the second experiment we performed parameter tying of the observation matrix. First, a 3-component mHMM was trained for 50 cycles on 34 sequences of 300 samples each, generated from a 3-component mHMM (2 hidden states, 3 observables); the sequences were distributed over 9 users. It was observed that good correspondence exists between the learned and the target A and B matrices (figure 2) and that all users are labelled correctly.

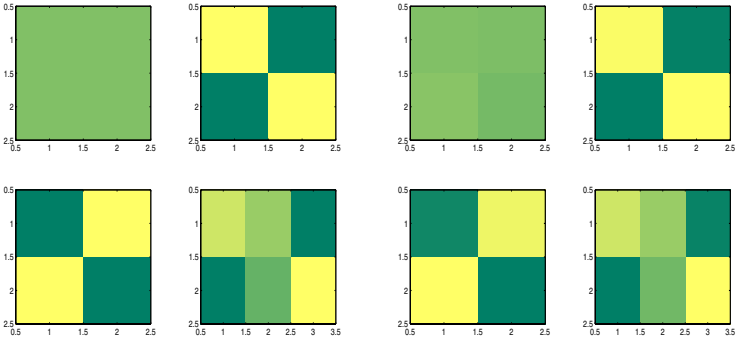


Fig. 2. Left 4 images: target A^1, A^2, A^3, B matrices; right 4 images: learned matrices

We then distributed 83 sequences from a 3-component mHMM (same parameter settings as above: 2 states, 3 observables, shared observation matrix) over 50 users (1 or 2 sequences each), where the sequence length was now much shorter (100 samples). For each user, an additional 6-D static data vector was drawn from a (spherical) Gaussian corresponding to each component; the Gaussians for different components were largely overlapping. We determined the number of erroneously labelled users with either static or dynamic data separately or jointly. The resulting error rates (dynamic: 4 %, static: 32 %, joint: 0 %) indicate that shorter sequences and a larger number of users give rise to erroneous labelling when using dynamical information only; combining dynamic with static information, however, leads to an improved user labelling.

3.2 Automatic Categorization of Web Logs

We applied an mHMM to (an excerpt of) a one-day log file from a large commercial web site in The Netherlands. The raw entries in the file were 'sessionized' [3] and irrelevant entries (like images) were removed. A training set was created out of clickstreams from 400 users. Then we trained a 4-component mHMM with 12 states and common observation matrix (with 134 different observables) for 500 cycles on the training set and we inspected the shared observation matrix B, the per-cluster transition matrices A^k and prior vectors Π^k.

We observed the shared observation matrix displayed in figure 3a. In this figure the horizontal axis denotes page-id, the vertical axis denotes state-id and a greyvalue at position (i, j) denotes the probability of observing page i in state j. The resulting page categorization was: 2,4,6,11: "shop info", 3,5: "start, customer/ corporate/ promotion", 1,8,12: "tools", 7,9,10: "search, download/ products". The 'semantic labeling' was done afterwards by manual inspection of the pages 'assigned to' a state. In the figure, the horizontal axis was reordered in such a way that pages that were assumed to belong to the same category have nearby page-id. We emphasize that this reordering was done *after* the learning phase, and only used to facilitate the presentation of the results. It can be seen

in the figure that similar pages (based on our manual assessment afterwards) are indeed assigned to particular categories.

Inspection of the learned state priors (vertical axis in figure 3b) for each of the 4 user types (horizontal axis) and the state transition matrices (one typical example is plotted in figure 4a) reveals that users are mainly assigned to 2 different user types with markedly different starting states. After inspection of the number of users that were assigned (with a large confidence) to each of the 4 user types, we noticed that the main clustering was w.r.t. user types '2' (app. 250 'general interest' users) and '3' (app. 150 'shop' users). All four transition matrices were fairly similar; the user labeling was apparently done based on the starting state. This can be understood, since 'shop' users are generally starting their session from a different page then the 'general interest' users. The other 2 components were dedicated to one particular user each.

We compared the results with an experiment with 6 assumed states and observed that three main page categories appear to be present: "shop info" (with its own starting page), "tools" and a "general interest" category, which is entered via the general starting page or via an initial search action. The

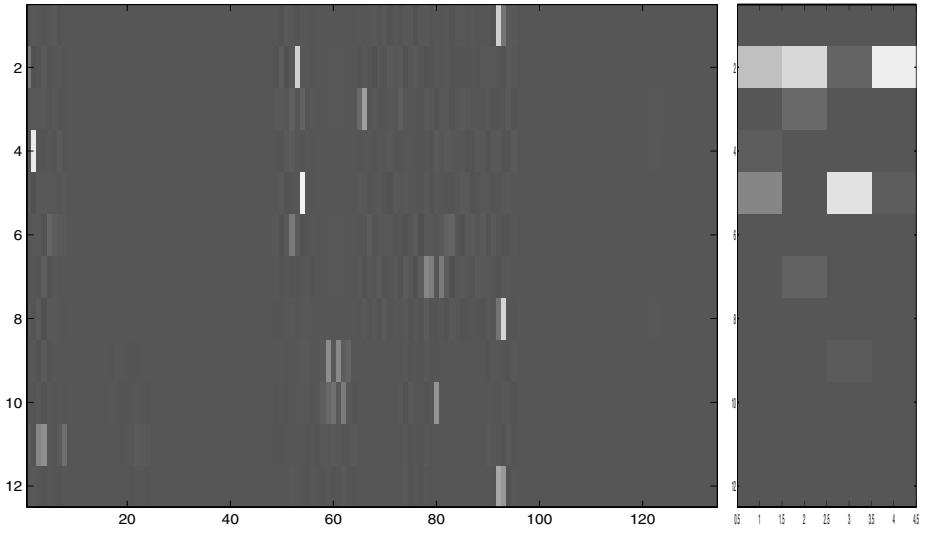


Fig. 3. a. (left): Learned 12-state observation matrix B; the shuffling of the page-ids is such that 1-51 = 'shop', 52-57 = 'general start & error', 58-64 = 'about & search', 65-77 = 'customer service', 78-91 = 'products', 92-94 = 'tools', 95-121 = 'download', 122-130 = 'corporate' and 130-134 = 'promotion'. It can be observed that 'shop' and 'tools' pages are assigned to distinct page categories. Furthermore, states tend to 'specialize' on certain 'general' page topics, e.g. 'customer, corporate, promotion' (from state 3) and 'search, download, products' (from states 7,9,10); b. (right): learned initial state vectors $\Pi^k, k = 1, \ldots, 4$. In both subfigures, large greyvalues denote high probabilities

transition matrices were again fairly similar (one instance is shown in figure 4b). In the figure, we reshuffled the state indices such that neighbouring state-ids are 'semantically similar'. This was done in order to match the ordering in the 'semantically similar' states of the 12-states experiment. Again the state-ids were reshuffled *after* the training procedure had taken place, in order to facilitate a visual comparison between the 12-state and the 6-state transition matrices. Indeed, the transition structure is comparable to that in the previous experiment (the transitions in figure 4a appear to be a 'zoomed in' version of the transitions in figure 4b).

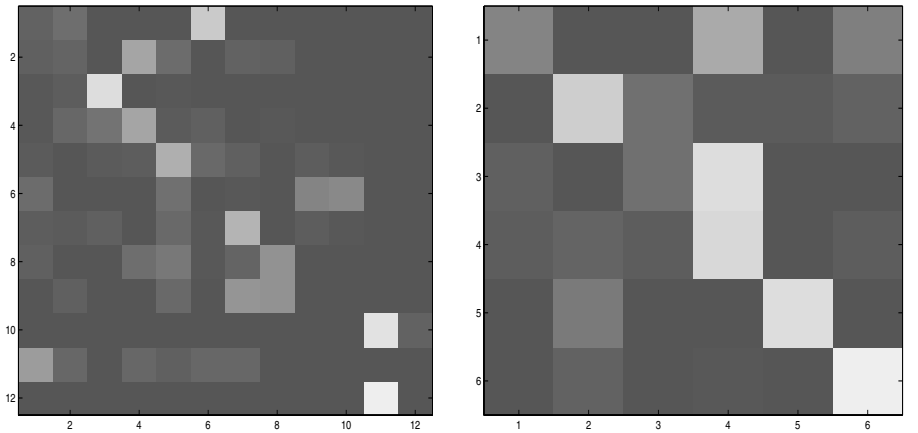


Fig. 4. a. (left): Learned 12-state transition matrix A^k; the shuffling of page-ids is such that 1-3 are the 'shop' category, 4-9 is the 'general' category and 10-12 is the 'tools' category; b. (right): learned 6-state transition matrix A^k; the shuffling of page-ids is such that 1,2 are the 'shop' category, 3,4,5 is the 'general' category and 6 is the 'tools' category

We verified our conjecture that 2 main clusters of users were present in the data by training an mHMM to half of the data from 3500 users with 2 and 3 mixture components respectively. In both cases we varied the number of states in the model. Then we computed the out-of-sample score on the remaining half of the data Y_{test} for each model $\Theta^{K,M}$ as

$$\text{score}(Y_{\text{test}}, K, M) = -\frac{\sum_i \log \sum_k \alpha_k P(Y^i_{\text{test}} | c_i = k, \Theta^{K,M}_k)}{\sum_{i,j} \text{length}(Y^{i,j}_{\text{test}})} \tag{11}$$

where $\Theta^{K,M}_k$ is the kth component of the mHMM with K components and M states. From our visual inspection of the learned categorization and user labelling we expected 2 user clusters and 3 categories to be present. However, it can be seen in figure 5 that there is no significant difference between the generalization performance of the 2 and 3 component models. Furthermore, including

more states leads to models with more predictive power. Including more than 14 states seems again to worsen the predictive power. This is an indication that it may be difficult to determine the 'meaningfulness' of the model purely on the basis of a criterion that measures the predictive quality. In [2] a similar problem was mentioned with respect to the choice of the number of components in a mixture of Markov chains. Comprising two disjunct components[4] into one may lead to models with more predictive power, but less interpretability. Note that our 'manual clustering and categorization' was based on the interpretability of the model.

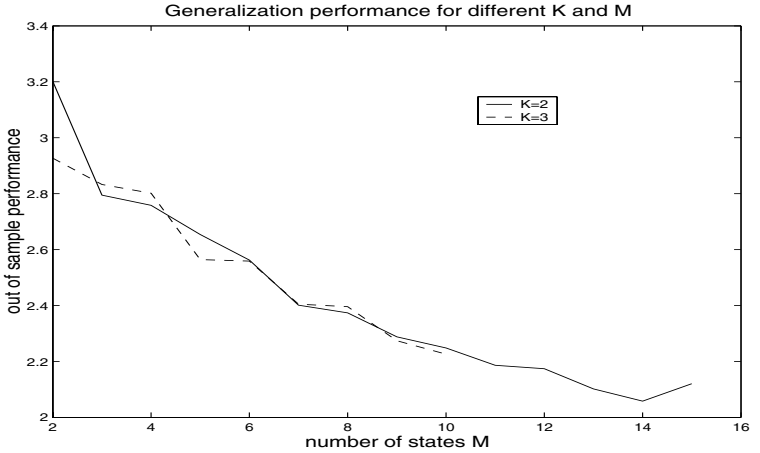


Fig. 5. Generalization performance of different mHMMs applied to web logs. There is no significant difference in performance between models with 2 or 3 mixture components. Including many states in an mHMM is preferable over including a few states. However, including more than 14 states seems to worsen generalization

4 Discussion

4.1 Discussion of Results

We noticed that two different choices for the number of categories underlying a set of clickstreams gave rise to semantically similar models for the data. This was concluded after a semantic analysis of the pages assigned to a state and by allowing that different page categories may consist of disjunct *groups* of states. In a practical setting, such a semantic analysis is not desirable. Then it becomes important to obtain a principled estimate of the number of states and the number of mixture components that is optimal for the data, e.g. by cross-validation or using

[4] E.g. comp. 1 = start in a, then go to b; comp. 2 = start in c, then go to d

model selection criteria. It is however not clear whether such a model selection that is based on predictive power will also lead to the most *interpretable* model, which is an issue of further research. The main aim of this paper, however, was to show that a meaningful page categorization may be learned simultaneously with the user labelling and inter-category transitions, making use of clickstreams and possibly static auxiliary data only.

4.2 Connection to Prior Work

Several authors have addressed (parts of) the web usage mining problem considered in this paper. The use of Markov models for modelling navigational patterns has been studied in [7,10]. In this work, each web page or page request corresponds to one state in a Markov model. In [10] this gave rise to successful predictions of HTTP-requests based on a stream of previous requests. However, the author mentioned the following limitations of the approach: lack of training data may give rise to low-quality models and cases with a large number of pages result in transition matrices that become intractable. In [7], the first-order Markov assumption was questioned, though a first-order model is considered more stable over a period of time than higher-order Markov models and typical clickstreams are usually short [4]. In our approach, the transitions are modelled at the level of page categories, which alleviates the intractability problem (transition matrices now scale with the number of page categories, which is expected to increase less than the number of pages). Moreover, we expect that the first-order Markov assumption will be more valid at the level of page categories[5].

Automatic clustering of user sessions and web pages were deemed the two interesting clustering tasks in the web usage domain [14]. Automatic categorization of web pages based on usage and content has been studied in [6]. After a content-based filtering of pages (e.g. by using a search engine), the connection structure between pages is used to identify hubs and authorities; the most relevant pages with respect to these criteria can then be categorized as belonging to this topic or query. Other approaches (e.g. see the overview in [11]) are based on a supervised content-based categorization of pages. In our approach we combine user clustering and page categorization in an unsupervised manner, using the link structure of the data only.

As stated before, our model is an extension of the work by [1]. If the states in our model are observed (e.g. when pages have been categorized manually) we have in fact a mixture of Markov chains, with update equations for Π, A and memberships c_{ik} that are similar to [1]. Moreover, we mention that an algorithm for training a mHMM has already been proposed in [12]. There it was stated that a mHMM is equivalent to one large 'overall' HMM with block-diagonal structure

[5] A reviewer added that many web sites are designed to promote the Markov assumption, i.e. each web page tries to motivate the user to stay at the web site and provides search tools and links to influence the user's next selection. This reviewer also expects that modelling state transitions at the level of page categories strengthens the Markov assumption

(which is due to the time-invariance of the cluster membership). We can see this by noting that $c_{ik} = P(c_i = k|Y^{i,j}, \Theta) = \sum_{\{x\} \in \mathcal{X}^k} P(X_t = x|Y^{i,j}, \Theta)$, where $\mathcal{X}^k$ denotes the states in the 'overall HMM' that correspond to cluster k. Despite the fact that memberships can be derived here from the 'overall HMM', the explicit formulation of memberships in our case offers a more intuitive way to combine clickstreams with static data.

4.3 Future Extensions

In future research, it may be useful to use the inter-page link structure of a web site as prior information about (im)possible inter-category transitions. For streamlining of a web site it may be necessary to use higher-order (hidden) Markov models. Moreover, user interests (and click behaviour) may change over time. Hence it may be interesting to analyze the user behaviour on, e.g., different days of the week. Finally, methods are needed to determine the optimal number of clusters and states for the problem at hand, though our results demonstrate that already suboptimal choices may lead to meaningful models.

5 Conclusion

We presented a method to cluster web surfers, based on their surfing patterns and additional information about the user. Since we learn the categorization of a web page along with the inter-category transition matrices and the cluster memberships, there is no need for a laborious prior labelling of pages. We demonstrated our method on artificial clickstreams and we retrieved a meaningful page categorization in an application to actual surfing patterns at a large commercial web site.

Acknowledgements. This work is supported by the Dutch Technology Foundation STW, project NNN.5321 "Graphical models for data mining". We thank KPN Research (current name: TNO Telecom) and KPN.com for supplying the web logs from the kpn.com web site.

References

1. I. Cadez, S. Gaffney, and P. Smyth. A general probabilistic framework for clustering individuals. Technical report, Univ. Calif., Irvine, March 2000.
2. I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of navigation patterns on a web site using model-based clustering. Technical report, Univ. Calif., Irvine, March 2000.
3. R. W. Cooley. *Web usage mining: discovery and application of interesting patterns from web data.* PhD thesis, University of Minnesota, USA, 2000.
4. B. A. Huberman, P. L T. Pirolli, J. E. Pitkow, and R. M. Lukose. Strong regularities in world wide web surfing. *Science,* (280):95–97, 1998.

5. M. I. Jordan, Z. Ghahramani, T. S. Jaakola, and L. K. Saul. An introduction to variational methods for graphical models. In *Learning in graphical models.* Kluwer Academic Publishers, 1998.
6. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. of 9th ACM-SIAM Symposium on Discrete Algorithms,* 1998.
7. M. Levene and G. Loizou. Computing the entropy of user navigation in the web. Technical report, Department of Computer Science, University College London, 1999.
8. L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE,* 77(2):257–285, February 1989.
9. M. Ramoni, P. Sebastiani, and P. Cohen. Bayesian clustering by dynamics. *Machine learning,* pages 91–121, 2002.
10. R. R. Sarukkai. Link prediction and path analysis using markov chains. In *Proceedings of the Ninth International World Wide Web Conference,* Amsterdam, 2000.
11. Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys,* 34(1):1–47, 2002.
12. P. Smyth. Clustering sequences with hidden markov models. In M.C. Mozer, M.I. Jordan, and T. Petsche, editors, *Advances in NIPS 9,* 1997.
13. P. Smyth. Probabilistic model-based clustering of multivariate and sequential data. In *Proc. of 7th Int. Workshop AI and Statistics,* pages 299–304, 1999.
14. J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations,* 1(2), 2000.

Web Usage Mining by Means of Multidimensional Sequence Alignment Methods

Birgit Hay, Geert Wets, and Koen Vanhoof

Limburg University, Data Analysis and Modeling Research Group
Universitaire Campus, gebouw D,
B-3590 Diepenbeek, Belgium
{birgit.hay,geert.wets,koen.vanhoof}@luc.ac.be

Abstract. In this article, a new algorithm called Multidimensional Sequence Alignment Method (MDSAM) is illustrated for mining navigation patterns on a web site. MDSAM examines sequences composed of several information types, such as visited pages and visiting time spent on pages. Besides, MDSAM handles large databases and uses heuristics to compute a multidimensional cost based on one-dimensional optimal trajectories. Empirical results show that MDSAM identifies profiles showing visited pages, visiting time spent on pages and the order in which pages are visited on a web site.

1 Introduction and Background

In general, Web Mining is a common term for three knowledge discovery domains: Web Content Mining, Web Structure Mining and Web Usage Mining [6, 30]. Web Content Mining is the process of extracting knowledge from the content of documents and their descriptions. Web Structure Mining is the process of inferring knowledge from the World Wide Web organization and links between pages in the Web. Finally, Web Usage Mining focuses on analyzing visiting information from logged data in order to extract previously unknown and interesting usage patterns [7]. In [8] Web Usage Mining is described as the application of data mining techniques on web access logs allowing management to optimize the site for the benefit of visitors. Besides, understanding and modeling visiting behavior may lead to strategies for web personalization and in general, strengthen competitive advantage. In this study we will focus on Web Usage Mining.

Fundamental ideas, concept definitions and surveys of Web (Usage) Mining are given in [5, 6, 9, 15, 17, 18, 24, 26, 30]. Studies show that, for mining visiting behavior on web sites, different techniques are used. For example, in [22] a tool for real-time knowledge discovery from users web page navigation called INSITE is presented. The system tracks users navigation through a web site and demonstrates real-time, scalable and adaptive clustering of navigation paths. A role-based recommendation engine allows for the web site to react to the user in real-time with customized information e.g. in target advertisement. Another technique is described in [25] where a Web Utilization Miner (WUM) discovers interesting navigation patterns. The system consists of two modules. The Aggregation Service prepares the logged data for mining while the MINT-Processor performs the mining. Besides, MINT supports the specification of criteria of statistical, structural and textual nature.

O.R. Zaïane et al. (Eds.): WEBKDD 2002, LNAI 2703, pp. 50–65, 2003.

Also, an innovative aggregated storage representation for the information in the web server log is exploited by WUM.

In [3] and [19] the order of elements in sequences is measured in mining visiting behavior on web sites. Yet, few studies in Web Usage Mining reveal *order-based information* of pages in navigation patterns. However, within Web Usage Mining research, mining visiting patterns based on the order of visited web pages offers important information for the purpose of supporting and increasing customer satisfaction. Such as optimization of the layout of the web site through structuring of page-links and web personalization techniques. Likewise, within our awareness, existing tools for *mining* two different information types like *visited web pages and time spent on pages* are hard to find. Despite the fact that visiting time on web pages is an interesting factor within Web Usage Mining studies [6, 7, 31] since it can be employed to measure user's interest in a page [23]. Therefore we will concentrate in this study on a measure for Web Usage Mining that well reflects structural information (represented by the order of elements) and handles two-dimensional sequences (providing two information types like visited pages- and visiting time). To this end, we will apply Multi dimensional Sequence Alignment Methods (MDSAM), which combines dynamic programming and genetic algorithms.

The article is organized as follows. First, Sequence Alignment Methods for one-dimensional and multidimensional datasets are described. Then, we illustrate our experiments using real log files of the web site Faculty Applied Economic Sciences. Finally the article is concluded and avenues for future research are given.

2 Sequence Alignment Methods

2.1 One-Dimensional Sequence Alignment Methods (SAM)

Description. One-dimensional Sequence Alignment Methods (SAM) are non-Euclidean distance measures between sequence pairs incorporating the order of elements in a sequence. SAM analyze sequences representing one information type and are applied within several research domains. The method, also called string edit distance, is used for sequence comparison within molecular biology and speech recognition by Sankoff and Kruskal [21]. A *sequence* is defined as a number of elements arranged successively or coming in succession one after the other. Likewise, in Mannila and Ronkainen [14] edit distance is described as a distance measure between event sequences. In traffic analysis studies the method is used to discover navigation patterns [12]. Finally, SAM is introduced in Web Usage Mining studies [11, 27].

In general, the distance or similarity between two sequences is reflected by the number of operations necessary to convert one sequence into the other. As a result, SAM distance measure is represented by a score. The higher/lower the score, the more/less effort it takes to equalize the sequences and the less/more similar sequences are. In addition, SAM scores for the following operations during equalization process: *Insertion* and *deletion* operations are applied to unique elements of source (first) and target (second) sequences; *reordering* operations are applied to common elements. Common elements appear in both compared sequences whereas unique elements

appear in either one of them. Moreover, a change in the order of elements is called a reordering operation. As a result, SAM represents the minimum cost for equalizing two sequences.

In particular, SAM distance measure between two sequences S_1 and S_2 is calculated using the following formula [21]:

$$d_{SAM}(S_1, S_2) = (w_d D + w_i I) + w_r R \quad (1)$$

where

d_{SAM} is the distance between two sequences S_1 and S_2, based on SAM;

w_d is the weight value for the deletion operations, a positive constant not equal to 0, determined by the researcher ($w_d > 0$);

w_i is the weight value for the insertion operations, a positive constant not equal to 0, determined by the researcher ($w_i > 0$);

D is the number of deletion operations;

I is the number of insertion operations;

R is the number of reordering operations;

w_r is the reordering weight, a positive constant not equal to 0, determined by the researcher ($w_r > 0$);

Equation (1) indicates that the score, represented by SAM distance measure between two sequences, consists of the costs for deleting and inserting unique elements and the costs for reordering common elements. Calculation of SAM distance measures between sequences is a combinatory problem. In practical applications, dynamic programming algorithms are used to resolve combinatory problems [12, 14, 28].

Example. To illustrate SAM, consider the following one-dimensional sequences s_1 (source sequence) and s_2 (target sequence). Both sequences represent sequentially ordered visited pages on a web site.

Suppose: $w_d = w_i = 1$ and $w_r = w_d + w_i$

s_1 {1, 4, 7, 8}

s_2 {1, 2, 3, 4, 5}

First identical elements having the same order of occurrence are defined. In the example above, page 1 is accessed before page 4 in both s_1 and s_2. Then, in order to equalize s_1 with s_2, unique elements 7 and 8 of s_1 are inserted into s_2 which gives the following sequences:

s_1 {1, 4, 7, 8}

s_2 {1, 2, 3, 4, 5, 7, 8}

The equalization process continues with deleting unique elements 2, 3 and 5 from s_2:

s_1 {1, 4, 7, 8}

s_2 {1, 4, 7, 8}

Finally, equalizing s_1 with s_2 took 2 insertion and 3 deletion operations, which gives us:

$d_{SAM}(s_1, s_2) = 5$

2.2 Multidimensional Sequence Alignment Methods (MDSAM)

Description. In order to capture similarity between sequences, including order-based information, for two or more dimensions, we introduce MDSAM [13]. For

multidimensional SAM distance measures, for example when sequences are compared with regard to pages and other information types such as time spent on pages, trajectories of operation sets are found using heuristics based on genetic algorithms [9, 16] to deal with combinatorial explosion of the problem. In other words, MDSAM optimizes the results of dynamic programming algorithms of one-dimensional optimal trajectories. Ultimately, MDSAM finds the set of operations inducing the possibly smallest sum of multidimensional operational costs.

In [13] MDSAM is described in detail. Figure 1 summarizes the MDSAM heuristic. For each sequence pair, a *trajectory* contains the operations on one information type to convert one sequence into the other. A set of trajectories represents a *population*. The population of the next generation is created by means of the following *genetic operators*: reproduction, crossover and mutation [16]. Reproduction is the process by which a program evaluates and copies strings according to the desired output. When crossover occurs, two strings exchange information that yields new combinations of scenarios. Mutation is a source of variation used to maintain diversity in a population of possible scenarios. The process integrates one-dimensional trajectories into multidimensional distance measures. Finally, the fitness of a population is calculated by the sum of the costs for deletions and insertions included in the multidimensional distance measures. The algorithm keeps on searching until improvement of multidimensional distance measures, during several runs, no longer occurs.

```
begin
  t=0                                   // t indicates the tth generation //
  no_improve = 0
  initialize E(t)                       // E(t) is the population //
  calculate C°(t)                       // C°(t) is the fitness of E(t) //
  Best_Fitness = C°(t)
  while not (no_improve >= convergence_rate) do
  begin
    select a genetic operator
    create E1(t) by selecting and copying a subset of E(t)
    t = t+1
    no_improve = no_improve +1
    create E1(t) by applying the selected genetic operator
  to E1(t-1)
    create E2(t) by selecting and copying a subset of E(t-1)
    create E(t) by summing E1(t) and E2(t)
    calculate C°(t)
     if C°(t) < Best_Fitness then
         Best_Fitness = C°(t)
        no_improve = 0
      end if
    end
end.
```

Fig. 1. Summarization of the MDSAM algorithm

Example. To illustrate MDSAM, consider the following two-dimensional sequences s_1 and s_2. The first attribute or information type (1, 4, 7, 8) in s_1 and (1, 2, 3, 4, 5) in s_2 indicates sequentially ordered visited pages whereas the second attribute (1, 4, 2, 3) in s_1 and (1, 2, 3, 4, 5) in s_2 corresponds to time spent (discrete information) on pages.
Suppose: $w_d = w_i = 1$ and $w_r = w_d + w_i$
s_1 {(1, 4, 7, 8); (1, 4, 2, 3)}
s_2 {(1, 2, 3, 4, 5); (1, 2, 3, 4, 5)}
Based on equation (1), MDSAM first employs dynamic programming algorithms to calculate optimal one-dimensional SAM distance measures for each attribute between sequences s_1 and s_2. This gives the following trajectories, showing the kind of operation (i=insertion and d=deletion), the position where the operation is applied in the sequence and the sequence that is affected by the operation:
attribute 1 (visited pages): $d_{SAM}(s_1, s_2) = 5$ trajectory = {$i3s_2$, $i4s_2$, $d2s_2$, $d3s_2$, $d5s_2$}
attribute 2 (time spent): $d_{SAM}(s_1, s_2) = 3$ trajectory = {$i2s_2$, $d4s_2$, $d5s_2$}
For example, for attribute 2, a SAM distance of 3 is measured after recognizing elements 1, followed by 2 and 3 as identities. Then a deletion of element 5 at position 5 in s_2 ($d5s_2$) is applied which gives the following sequences:
s_1 {1, 4, 2, 3}
s_2 {1, 2, 3, 4}
Equalization continues with a reordering operation of element 4. In a trajectory, reordering is represented by a deletion and an insertion operation. In this example, element 4 is inserted at position 2 in s_2 ($i2s_2$) and deleted at position 4 from s_2 ($d4s_2$):
s_1 {1, 4, 2, 3}
s_2 {1, 4, 2, 3}
Finally, equalizing s_1 with s_2 regarding the second attribute took 1 insertion and 2 deletion operations, which gives us:
attribute 2 (time spent): $d_{SAM}(s_1, s_2) = 3$

However, a simple sum of the results of one-dimensional SAM distance measures does not capture inter-attribute relationships between sequences. For example, because in s_1 less pages (attribute 1) are visited compared to s_2, this will automatically lead to an operation with regard to visiting time (attribute 2). Therefore, to integrate two one-dimensional trajectories into a multidimensional trajectory, MDSAM bundles operations of the same kind that are applied to the same position between attributes into a single operation. Following the example above, MDSAM distance between s_1 and s_2 equals 7 instead of 8 because $d5s_2$ is found in both trajectories. Unfortunately this alignment procedure does not always provide the optimum solution. For example, a non-optimal trajectory for attribute 2 like {$i3s_2$, $i4s_2$, $d2s_2$, $d3s_2$, $d5s_2$}, after identifying elements 1 and 4 as identities, induces a smaller MDSAM distance measure of 5, compared to 7, due to bundling up five operations of attribute 1 and 2. Therefore, all possible trajectories must be evaluated, which gives the following set of trajectories:
attribute 1 (visited pages): $d_{SAM}(s_1, s_2) = 5$ trajectory = {$i3s_2$, $i4s_2$, $d2s_2$, $d3s_2$, $d5s_2$}
$d_{SAM}(s_1, s_2) = 7$ trajectory = {$i2s_2$, $i3s_2$, $i4s_2$, $d2s_2$, $d3s_2$, $d4s_2$, $d5s_2$}
…
attribute 2 (time spent): $d_{SAM}(s_1, s_2) = 3$ trajectory = {$i2s_2$, $d4s_2$, $d5s_2$}

$d_{SAM}(s_1, s_2) = 5$ trajectory = { $i3s_2$, $i4s_2$, $d2s_2$, $d3s_2$, $d5s_2$}
…

Following optimum alignments of each attribute, MDSAM continues using genetic algorithms to find the possibly smallest sum of multidimensional operation costs. The best trajectory sets are selected for the next generation until, during several runs, the costs are not lowered. In this example MDSAM between s_1 and s_2 equals 5, presenting the following trajectory:

attribute 1 (visited pages) + attribute 2 (time spent):

$d_{MDSAM}(s_1, s_2) = 5$ trajectory = {$i3s_2$, $i4s_2$, $d2s_2$, $d3s_2$, $d5s_2$}

3 Experiments

Web Usage Mining discovers patterns in behavior of visitors to a web site allowing management to optimize the site for the benefit of visitors. An important part of the process of recognizing patterns in data is cluster discovery [8], which is illustrated by our approach of Web Usage Mining, presented in figure 2. The process includes three main steps [5, 8]: data pre-processing, data mining and pattern evaluation. Output of each step is used as input for each subsequent step. In order to get a general view about what is going on at a web site, the purpose of the experiment is mining for navigation patterns providing general information about visited pages and visiting times.

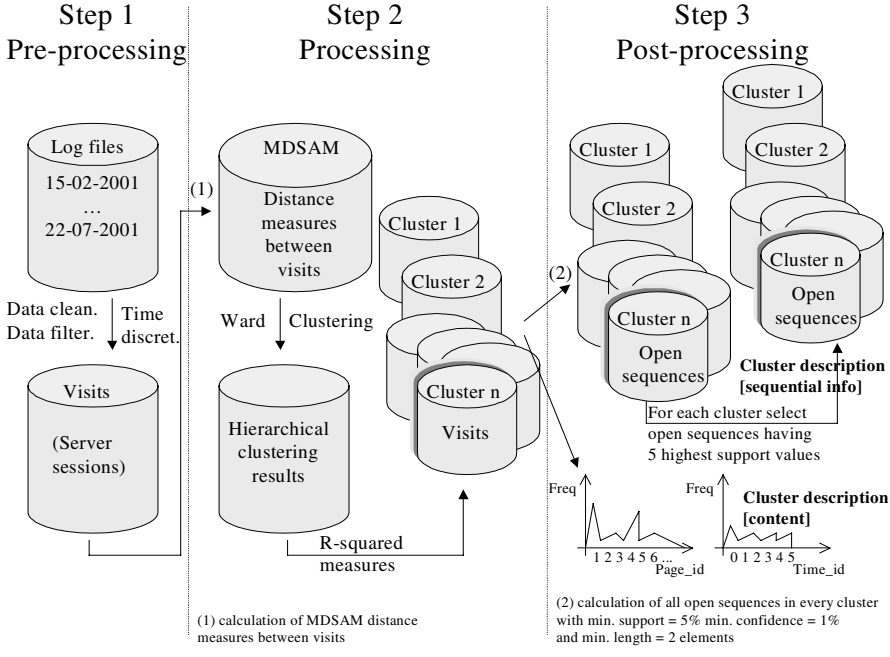


Fig. 2. Web Usage Mining Process

3.1 Step 1: Data Pre-processing

For this project, web access logs of the web site Faculty Applied Economic Sciences are analyzed. The log files register navigation data from 15/02/2001 until 22/07/2001. In order to analyze visiting behavior on a web site, sessions of web-click stream data must be defined. A *server session* or *visit* is defined as the click-stream of page-views for a single visit of a user to a web site [5]. In this article, we will use server session or visit interchangeably. For this study, server sessions are composed of two different information types: page- and time information. First, data stored in log files are cleaned in such a way that only URL page requests of the form 'GET...html' are maintained. Then, a unique code is given to each distinct ip-address and URL. Sessions are identified using a threshold of 30 minutes viewing time. This means that, with the same ip-address, a new session starts when the time between two page requests is more than 30 minutes. In general, a session is created when a new ip-address is met in the log file. Finally, a filtering method is invoked on the sequences in order to identify visitors using the same ip-address. We remark that, the focus of the work is on (MD)SAM and less attention is paid to identification of unique users and visits. Further research will involve application of heuristics for user identification and other methods that rely on user cooperation.
Visiting time information is discredited into alphanumerical characters presented in table 1. For example, if a page is visited for less than 11 seconds an alphanumerical character of 0 is used in the analysis. In order to evaluate the final results of the analysis a percentage of the number of clicks for each category is given in the third column of table 1. The data show that 38.25% of the pages are visited within a time period of 1 to 5 minutes. For this experiment we define *visit time* (or viewing time) of a page as the time difference between page request and subsequent page request [23]. The approach of using time windows is illustrated in [7] for transaction identification. Likewise in [6] time windows are used to find common characteristics of users that visited a particular page within the time period $[t_1, t_2]$. Missing values with regard to time information of the last page of a server session are substituted by the average visiting time of that particular page [29]. We remark that, in this study, we focus on information retrieval based on MDSAM and less attention is paid to computing accurate viewing time.

Table 1. Discredited visiting time information

Time Window (seconds)	Category	Distribution (%)
≤10	0	24.68
11-30	1	19.60
31-60	2	8.80
61-300	3	38.25
>300	4	8.67

Eventually, server sessions are built in the form of session-id, {(<page-id>); (<time-id>)} representing consecutive pages requested by the same user with corresponding visiting time information. For example, a visit s_1, {(1, 3, 5); (0, 0, 3)} tells us that a user enters the web site through page 1, spends a short period of time on

that page (less than 11 seconds) and then visits page 3 for a short time. In the end, the user proceeds to page 5, spends more time on that page (between 1 and 5 minutes) and exits the web site.

3.2 Step 2: Data Mining

In the second step of the Web Usage Mining process, MDSAM distance measures are calculated between visits. Consequently, a similarity matrix holding pair wise multidimensional distance measures between visits is used as distance measure for clustering. Because this study is focused on MDSAM, no special attention is paid to the clustering method. Therefore, a simple method for hierarchical clustering like Ward's method [10] is invoked on the similarity matrix. Further research will include exploration of other clustering methods. In order to define an optimal solution for the number of clusters, r-squared is used as a goodness-of-fit measure during cluster processing. *R-squared* equals the proportion of variation explained by the model [10] and ranges in value from 0 to 1. Obviously, the level of r-squared increases with the number of clusters. On the other hand, cluster analysis will become economically unfeasible because of computational demands if too many clusters need to be analyzed. Therefore, a trade off between number of clusters and model fit must be defined. Small values of r-squared indicate that the model does not fit the data well, whereas measures of 0.6 and higher are considered acceptable [10]. Ultimately we will define a stop criterion when the incremental values of r-squared flatten out if additional clusters are formed. This means that for this experiment 6 clusters are defined with r-squared equal to 0.9.

3.3 Step 3: Pattern Evaluation

Capri's Open Sequences. In the third step software Capri [4] is applied to every cluster in order to compute two-dimensional open sequences to define navigation patterns, presenting page- and time information. In Büchner et al. [2] open sequences are used for discovering structural information within navigation patterns. Sequences with the same elements occurring in the same order and irrelevant of the positions of the elements are called *open sequences*. When all open sequences with minimum support of 5%, minimum confidence of 1% and minimum length of 2 elements are calculated for every cluster, open sequences having 5 highest support values are selected for cluster description. *Support* is specified as the number of cases within a cluster presenting the open sequence divided by the total number of cases in that cluster. *Confidence* expresses the probability that, if a case in a cluster contains all but the last page (in respective order) and corresponding time information of the open sequence, the case will also hold the last page and corresponding time information of the open sequence. For example, for an open sequence {(1, 68); (0, 0)} consisting of 2 information types, having each 2 elements, presenting page and time information, a confidence of 42% says that, of all the visitors going to page 1 and viewing page 1 during less than 11 seconds, 42% proceeds to page 68 and stays on this page for less than 11 seconds. In figure 3 graphical presentations of the results are given. The web

site structure is printed in each cluster, showing page numbers between brackets and links between pages by means of solid arrows. For example, page 68 can be accessed through page 1 and vice versa. A visitor can also go from page 43 to page 55. However, the other way around is not possible here. The first four pages at the upper level in each cluster presentation constitute the main pages of the web site, with (1) representing the root directory or home page. In order to avoid complex drawings of arrays making the figures unclear, two horizontal lines are printed which indicate links from pages 43, 49, 65 and 71 to main pages 1, 2 and 9. Also page 55 above the second horizontal line is linked to pages 1, 2, 9 and 68. For evaluation purposes, distribution of sequences is given in the upper right corner of every cluster. For example, 15.41% of the server sessions in the input file are grouped in cluster 1.

In each cluster, visitor navigations are drawn by dashed arrows in figure 3. For every pattern, support (s) and confidence (c) values are written next to the dashed arrow. Time information is written between brackets alongside the dashed arrow. The first number indicates visiting time of the page where the arrow begins whereas the second number refers to visiting time of the page where the arrow ends or points at. For example, in cluster 1, 19.01% of the visits went from page 1 to page 68. Furthermore, both pages were visited during a very short time period of less than 11 seconds, indicated by time information of 0. A confidence value of 42% shows that, of all the visitors that stayed less than 11 seconds on page 1, 42% will proceed to page 68 and stay on that page for less than 11 seconds. Following cluster 3, 15.30% of the server sessions show twice visits to page 68, indicated by pattern {(68, 68); (0, 0)}. Also 10.27% of the visits hold {68, 68); (0, 1)}. A circle around page 68 illustrates this. However, navigations are not shown in cluster 2 because no open sequences are found. During a second run the minimum support level is lowered to 1% and Capri searched for all possible open sequences. Still, no open sequences are found. This indicates that cluster 2 groups special cases. Further analysis reveals that cluster 2 contains exceptional visits, shown by very long or very short sequences, to pages at lower levels of the web site. Visit {(68, 65, 56, 58, 59, 60, 61, 63, 64, 43, 39, 40, 26, 28); (0, 1, 1, 0, 1, 0, 2, 0, 1, 2, 3, 0, 0, 2)} and visit {(8, 11); (0, 3)} are two illustrations of sequences in cluster 2.

Towards cluster analysis the following observations are made. In figure 3 it may occur that, at first sight, the same open sequences presenting page- and time information describe different clusters. For example, open sequence {(68, 68); (0, 0)} characterizes cluster 3 and 4. However, sequences in cluster 3 holding {(68, 68); (0, 0)} also hold {(68, 43); (0, 1)} and/or {(68, 71); (0, 3)} and/or {(68, 71); (1, 3)} or are very similar to these patterns. Besides, sequences in cluster 4 holding {(68, 68); (0, 0)} have also navigations towards page 65. This implies that sequences grouped within the same cluster have the same characteristics towards visited pages, order of pages and visiting time information whereas sequences of different clusters have quite different features.

Ultimately, in our analysis, alignment methods precede open sequences. If open sequences were searched first, followed by using alignment methods, we would be clustering open sequences instead of server sessions and might lose information embedded in server sessions. Besides, applying alignment methods on open sequences instead of server sessions will treat open sequences with relatively high support and confidence the same way as open sequences with relatively low support

and confidence. In the end, clusters may end up holding open sequences with relatively low support and confidence values. This means that a cutoff value needs to be defined to extract open sequences with high support and/or confidence values. Defining a cutoff value before the mining process takes place means that only part of the data will be analyzed and valuable information might be lost. For this reason, we apply sequence alignment methods before open sequences and not the other way around.

Frequency Graphs. In order to provide a general overview of the content of every cluster, two frequency graphs present visited web pages and time information. In figures 4 and 5, for every cluster, the first graph specifies on the horizontal axis the code of the web page with frequency values presented vertically. The second graph labels discrete time information horizontally with respective frequency values vertically.

3.4 Interpretation of Clusters and Deployment Possibilities

Extracted profiles of MDSAM are given in table 2. Six profiles are extracted. First, general profiles of clicks to main pages are shown: from 'root' to 'services', 'education' and 'information', and from 'education' to 'information'. All pages are visited for a very short time (i.e. less than 11 seconds) except for 'information'. Here, visitors generally stay longer, recording a visiting time between 1 and 5 minutes. A second profile recognizes clicks to pages at lower levels of the web site structure. Visits are exceptionally long or short, which means that on the one hand many pages are visited and on the other hand visits consist of only one or two page-clicks. Mostly, time information between 1 and 5 minutes is shown. A third profile describes patterns from main page 'education' to pages at a lower level: 'study-elements of Economic Engineering' and 'news and examinations'. Most pages are visited during less than 11 seconds while people stay longer on 'news and examinations'. Another profile shows clicks from main page 'education' to a lower level 'study-elements of Applied Economic Sciences'. Three visiting groups spend respectively less than 11 seconds, between 11 and 30 seconds and between 1 and 5 minutes on 'study-elements of Applied Economic Sciences'. In a fifth profile visitors who click only to main page 'root' are grouped. This pattern occurs because staff members of the faculty use web mail through the 'root' page of the web site. Finally, a sixth profile identifies visits containing repeated clicks to 'education' without navigations to other pages of the web site. Ultimately, sequences grouped within the same cluster have the same characteristics towards visited pages, order of pages and visiting time information whereas sequences of different clusters have quite different features.

The results of this study may be deployed as follows. Web developers may verify whether visitors use the web site conform with its structure. For example, navigation pages should lead visitors to content pages and should be visited during a very short time. In cluster 3 (re. figure 3), visitors use page 68 'education' to proceed to page 71 'news and examinations', which is a content page within the structure of the web site. Time information shows that most of the visitors stay less than 11 seconds on page 68 and more than one minute on page 71, indicating that visitors use the web site

conform with the intentions of the web developer. Yet, cluster 4 shows that page 65 'study-elements of Applied Economic Sciences', which is a navigation page leading to content pages 55 and 57, is visited within three different time categories of respectively 0, 1 and 3. This means that the 'road' to content pages 55 and 57 is not followed by the visitors and therefore suggests optimizing part of the structure for the benefit and convenience of users. For example, moving pages 55 and 57 to a higher level of the web site structure. Also, since page 65 is not used as a navigation page by most of the visitors, people might not directly find what they are looking for. This might suggest splitting page 65 up into two or several pages.

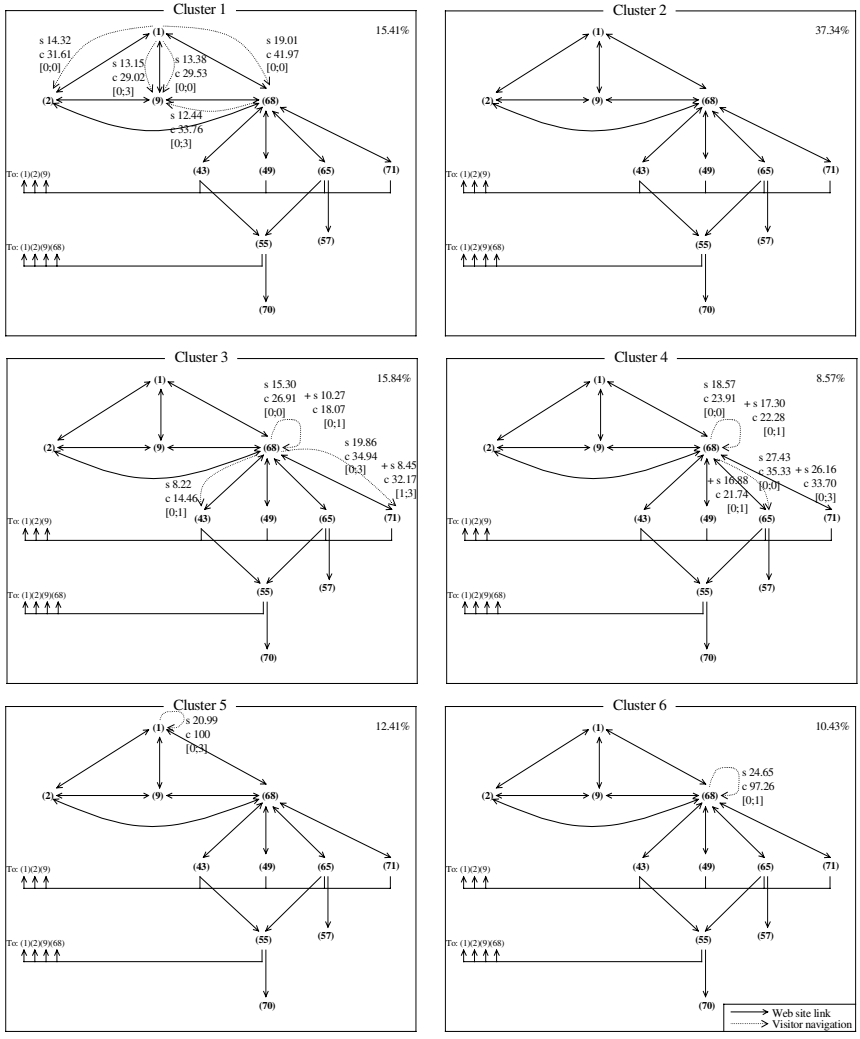


Fig. 3. Navigation patterns based on two dimensions (page- and time information) grouped within 6 clusters

4 Conclusions and Future Research

In order to provide a general view about what is going on at a web site, the purpose of this study is mining for navigation patterns providing general order-based information about visited pages and visiting times. In this article, Multidimensional Sequence Alignment Methods (MDSAM) are applied to web usage data in order to extract multidimensional profiles from visiting behavior within Web Usage Mining studies. MDSAM is a non-Euclidean pair wise distance measure and compares sequences consisting of minimum two attributes or information types.

Table 2. Profiles of clusters

Cluster	Profile
1	Clicks to main pages 'root' (page 1), 'services' (page 2), 'education' (page 68) and 'information' (page 9). Time information of less than 11 seconds appears except for page 9. If this page is visited after page 68 or page 1, a visiting time between 1 and 5 minutes may occur.
2	Exceptional visits to pages at lower levels of the web site structure. Sequences consist of many pages or represent only one or two page clicks.
3	Repeated clicks to main page 'education' (page 68), 'study-elements of Economic Engineering' (page 43) and 'news and examinations' (page 71). People stay longer on page 71 compared to other pages.
4	Repeated clicks to main page 'education' (page 68) and 'study-elements of Applied Economic Sciences' (page 65). Three visiting groups spend respectively less than 11 seconds, between 11 and 30 seconds and between 1 and 5 minutes on page 65.
5	Repeated clicks to 'root' (page 1). The first click illustrates a very short visiting time while the second one is visited during 1 to 5 minutes.
6	Repeated clicks to 'education' (page 68) without navigations to other pages of the web site.

Experiments on a real data set from the web site Faculty Applied Economic Sciences show that clustering server sessions by means of MDSAM identifies six profiles providing order-based information of two attributes: visited pages and visiting time information. Server sessions that are grouped in the same cluster have the same characteristics towards visited pages, order of pages and visiting time information whereas server sessions of different clusters have quite different features.

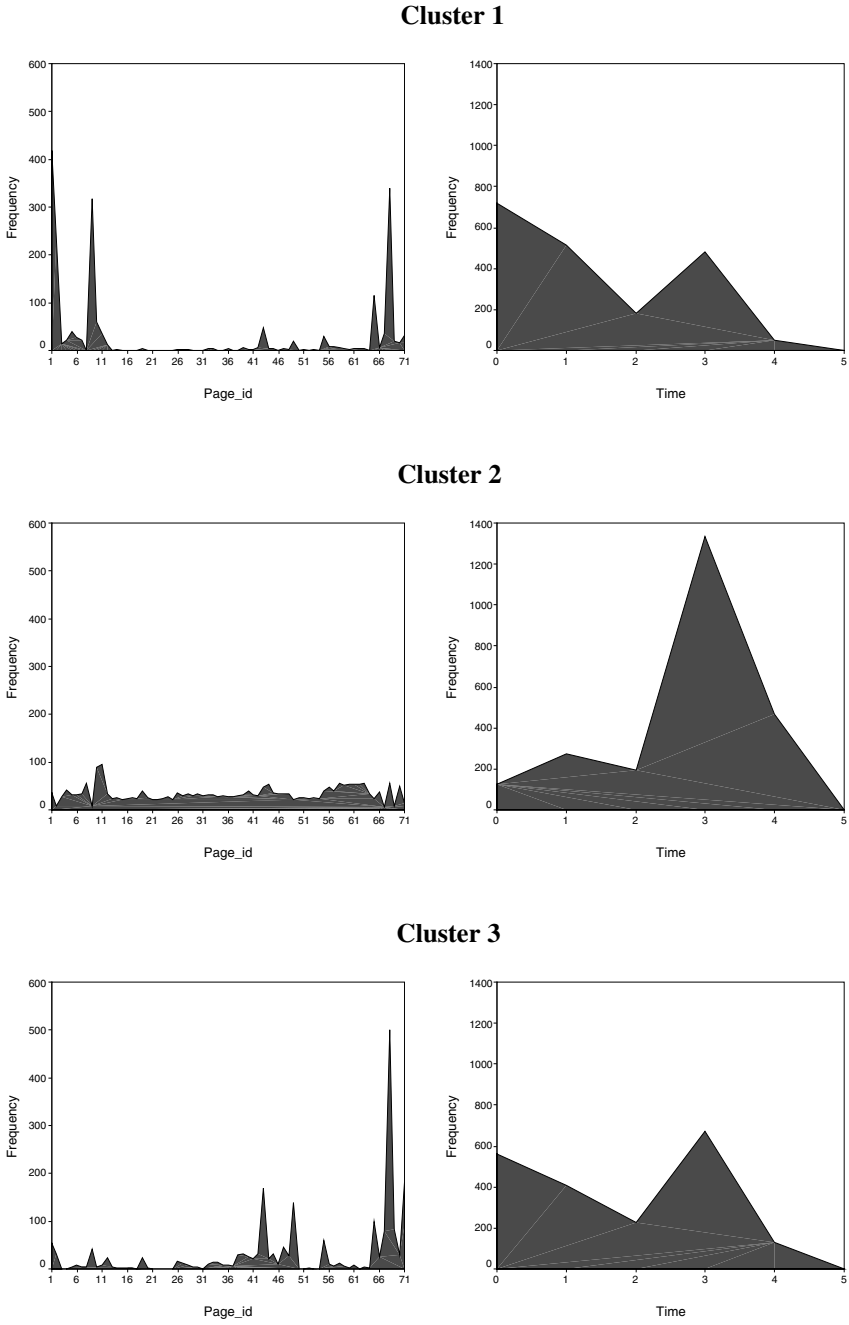


Fig. 4. Frequency graphs presenting the content of cluster 1, 2 and 3

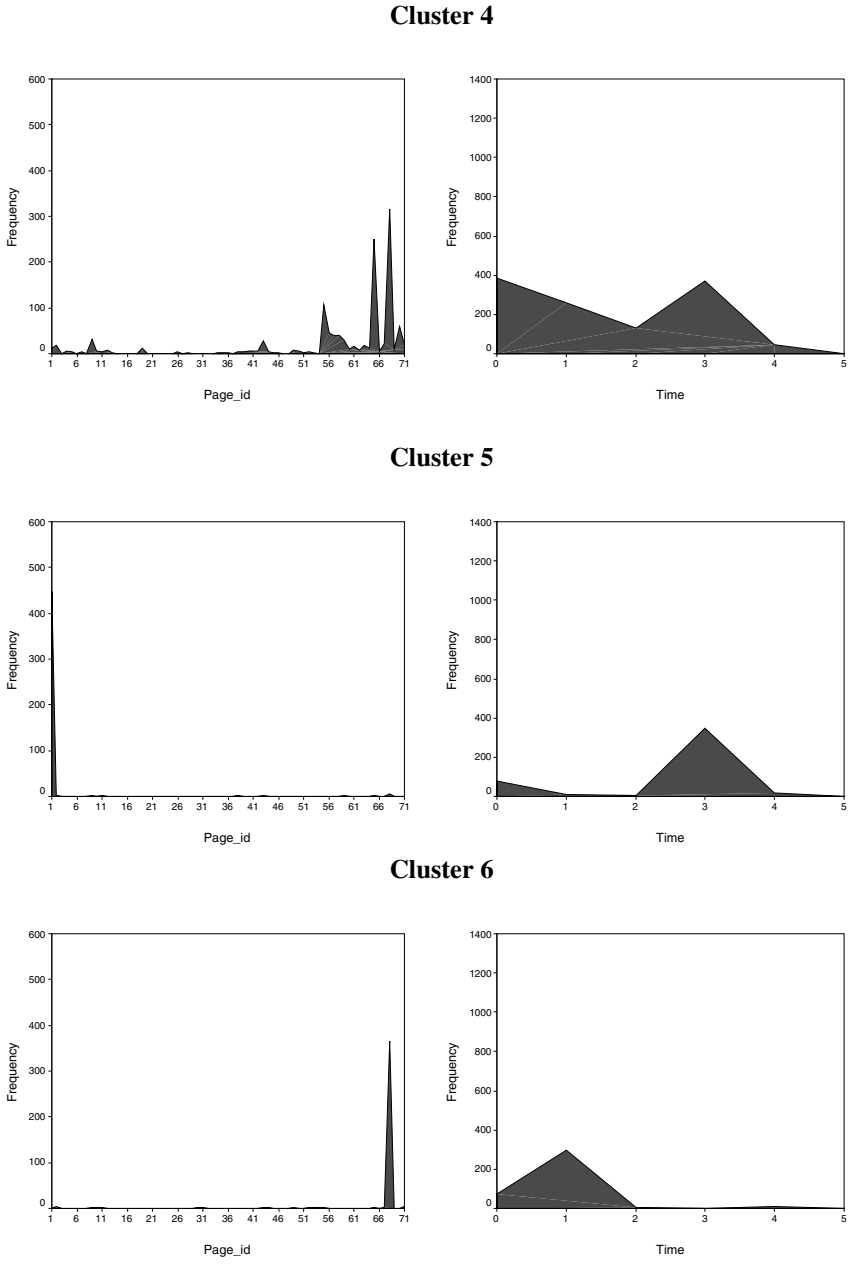


Fig. 5. Frequency graphs presenting the content of cluster 4, 5 and 6

To confirm our finding, the method should also be examined using other clustering algorithms (e.g. mixture models). Likewise several criteria to find the optimal clustering solution should be evaluated. In addition, the strength of MDSAM should further be examined using different sets of parameters. For example, analyzing

the effect of a change in insertion, deletion or reordering weight on the clustering results. On the other hand, the method should be extended by means of integration with the web site tree structure (for example using token strings) and inclusion of shi-squared statistical tests as a factor for measuring interestingness. Another topic for further research concerns the scalability of MDSAM, since the clustering is based on an initial pair-wise similarity matrix. Finally, the effects of using heuristics for user identification and other methods that rely on user cooperation on the final results are to be examined.

References

1. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. *Proceedings Conference ACM SIGMOD on Management of Data (ACM SIGMOD CMD '98),* pp. 94–104, Seattle, WA, Jun. (1998)
2. Büchner, A.G., Baumgarten, M., Anand, S.S., Mulvenna M.D., Highes, J.G.: Navigation Pattern Discovery from Internet Data. *Proceedings Workshop ACM on Web Usage Analysis and User Profiling (ACM WEBKDD '99),* pp. 25–30, San Diego, CA, Aug. (1999)
3. Cadez, I., Heckerman, D., Meek, C., Smyth, P., White, S.: Visualization of Navigation Patterns on a Web Site Using Model Based Clustering. Technical Report MSR-TR-2000-18*,* Microsoft Research (2000)
4. Capri: Generic sequence discovery product. http://www.mineit.com/products (2001)
5. Cooley R.: Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data. Ph. D. Thesis*,* University of Minnesota, http://www.users.cs.umn.edu/~cooley/pubs.html (2000)
6. Cooley, R., Mobasher, B., Srivastava, J.: Web Mining: Information and Pattern Discovery on the World Wide Web. A survey paper. *Proceedings 9th IEEE Conference on Tools with Artificial Intelligence (ICTAI '97),* Newport Beach, CA, Nov. (1997)
7. Cooley, R., Mobasher, B., Srivastava, J.: Data Preparation for Mining World Wide Web Browsing Patterns. Knowledge and Information Systems, Vol. 1, No. 1. (1999) 5–32
8. Foss, A., Weinan, W., Zaïane, O. R.: A Non-Parametric Approach to Web Log Analysis. *Proceedings of Workshop on Web Mining in First International SIAM Conference on Data Mining (SDM '01),* pp. 41–50, Chicago, IL, Apr. (2001)
9. Goldberg, D.E.: Genetic algorithms in search, optimization and machine learning. Reading, Addison-Wesley, MA (1989)
10. Hair, J., Andersen, R., Tatham, R., Black, W.: Multivariate Data Analysis. Prentice Hall, New Jersey (1998)
11. Hay, B., Wets, G., Vanhoof, K.: Clustering Navigation Patterns on a Website Using a Sequence Alignment Method. *Proceedings 17th Conference on Artificial Intelligence, Intelligent Techniques for Web Personalization (IJCAI '01),* pp. 1–6, Seattle, WA, Aug. (2001)
12. Joh, C.H., Arentze, T.A., Timmermans, H.J.P.: A position-sensitive sequence alignment method illustrated for space-time activity-diary data. Environment and Planning A, Vol. 33, No. 2. (2001) 313–338
13. Joh, C.H., Arentze, T.A., Timmermans, H.J.P.: Multidimensional Sequence Alignment Methods for Activity-Travel Pattern Analysis: a comparison of Dynamic Programming and Genetic Algorithms. Geographical Analysis, Vol. 33, No. 3. (2001) 247–270.

14. Mannila, H., Ronkainen, P.: Similarity of event sequences. *Proceedings 4th Workshop on Temporal Representation and Reasoning (TIME '97),* pp. 136–139, Daytona Beach, Florida, May (1997)
15. Masand, B., Spiliopoulou, M.: Advances in Web Usage Mining and User Profiling. *Proceedings Workshop ACM on Web Usage Analysis and User Profiling (ACM WEBKDD '99)* and LNAI, Vol. 1836, Springer Verlag, Jul. (2000)
16. Mena, J.: Data Mining Your Website. Digital Press, Boston (1999)
17. Mobasher, B., Jain, N., Han, E., Srivastava, J.: Web Mining: Pattern discovery from World Wide Web transactions. Technical Report TR 96–050, University of Minnesota (1996)
18. Mulvenna, M.D., Anand, S.S., Büchner, A.G.: Personalization on the Net using Web mining: introduction. Communications of the ACM, Vol. 43, No. 8. (2000) 122–125
19. Nasraoui, O., Krishnapuram, R., Anupam, J.: Mining Web Access Logs Using a Fuzzy Relational Clustering Algorithm Based on a Robust Estimator. Proceedings 8th World Wide Web Conference (WWW8 '99), Toronto, Canada, May (1999)
20. Piatetsky-Shapiro, G., Fayyad, U., Smith, P.: From data mining to knowledge discovery: An overview. In: Fayyad, U.M., Piatetsky-Shapiro, G., Smith, P., Uthurusamy, R. (eds.): Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press (1996) 1–35
21. Sankoff, D., Kruskal, J.B.: Time warps, string edits and macromolecules: the theory and practice of sequence comparison. Reading, Addison Wesley, Mass. (1983)
22. Shahabi, C., Faisal, A., Kashani, F.B., Faruque, J.: INSITE: A Tool for interpreting Users? Interaction with a Web Space. *Proceedings 26th Conference on Very Large Databases (VLDB '00),* pp. 635–638, Egypt, Caïro, Sept. (2000)
23. Shahabi, C., Zarkesh, A., Adibi, J., Shah, V.: Knowledge discovery from users Web-page navigation. *Proceedings 7th Workshop IEEE on Research Issues in Data Engineering,* pp. 20–31, UK, Birmingham, Apr. (1997)
24. Spiliopoulou, M.: Web Usage Mining: Data Mining über die Nutzung des Web. In: Hippner, Ulrich, Küsters, Meyer, Wilke (eds): Handbuch Data Mining im Marketing, chapter 13. Vieweg (2000)
25. Spiliopoulou, M., Faulstich, L.: WUM: a Tool for Web Utilization Analysis. *Proceedings Workshop of World Wide Web and Databases (WebDB '98),* pp. 84–103, Spain, Valencia, March 1998. Extended version in LNCS, pp. 84–103, Vol. 1590 (1998)
26. Srivastava, J., Cooley, R., Deshpande, M., Tan, P-N: Web usage mining: discovery and applications of usage patterns from web data. ACM SIGKDD Explorations, Vol. 1, No. 2 (2000) 12–23
27. Wang, W., Zaïane, O.R.: Clustering Web Sessions by Sequence Alignment. *Proceedings 3rd Workshop on Management of Information on the Web in conjunction with 13th International Conference on Database and Expert Systems Applications DEXA,* Aix en Provence, France, Sept. (2002)
28. Wilson, W.C.: Activity pattern analysis by means of Sequence Alignment Methods. Environment and Planning, Vol. A, No. 30 (1998) 1017–1038
29. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Fransisco (2000)
30. Zaïane, O.R.: Conference Tutorial Notes: Web Mining: Concepts, Practices and Research. *Proceedings XIV Brazilian Symposium on Databases (SDBD'2000),* Promoted by the Brazilian Computer Society in cooperation with ACM SIGMOD, pp. 410–474, Brazil, João Pessoa, Paraïba, Oct. (2001)
31. Zaïane, O.R., Xin, M., Han, J.: Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. *Proceedings of Advances in Digital Libraries, IEEE,* pp. 19–29, Santa Barbara, Apr. (1998)

A Customizable Behavior Model for Temporal Prediction of Web User Sequences

Enrique Frías-Martínez and Vijay Karamcheti

Courant Institute of Mathematical Sciences, New York University
715 Broadway, New York NY 10003 USA
{frias,vijayk}@cs.nyu.edu

Abstract. One of the important Internet challenges in coming years will be the introduction of intelligent services and the creation of a more personalized environment for users. A key prerequisite for such services is the modeling of user behavior and a natural starting place for this are Web logs. In this paper we propose a model for predicting sequences of user accesses which is distinguished by two elements: it is customizable and it reflects sequentiality. Customizable, in this context, means that the proposed model can be adapted to the characteristics of the server to more accurately capture its behavior. The concept of sequentiality in our model consists of three elements: (1) preservation of the sequence of the click stream in the antecedent, (2) preservation of the sequence of the click stream in the consequent and (3) a measure of the gap between the antecedent and the consequent in terms of the number of user clicks.

1 Introduction

Since Etzioni [12] first proposed the term "Web Mining" a lot of research has been done in the area of analyzing web servers logs to detect patterns and to find user characteristics. One topic that has received a lot of attention is modeling the behavior of web users, in other words, being able to predict the requests of users. Having such a model of user behavior has made possible the implementation of a great variety of intelligent services. Some examples of these services include: redesigning of web sites [24], personalization of e-commerce sites [27], recommendation of pages [18], construction of web pages in real-time [23], adaptation of web pages for wireless devices [4], improvement of web search engines, and prefetching [11][19][30][31].

The main techniques traditionally used for modeling user's patterns are clustering and association rules. Clustering is a technique that can be used to group similar browsing patterns or to divide the web pages of a site into groups that are accessed together. Association rules detect frequent patterns among transactions. Despite their popularity, these two approaches produce systems which lack two important characteristics of web user access: sequentiality and temporality. In this context sequentiality implies reflecting the order of the requests of the user, and temporality refers to being able to capture when the predicted actions are actually going to happen.

O.R. Zaïane et al. (Eds.): WEBKDD 2002, LNAI 2703, pp. 66–85, 2003.

In this paper we present a model that constructs sequential rules which, unlike clustering and association rules, capture the sequentiality and temporality in which web pages are visited. These rules are defined by an antecedent and a consequent. Both antecedent and consequent are subsequences of a session. In order to preserve sequentiality, the rules maintain the sequence of the click stream of the antecedent and of the consequent. The concept of temporality is reflected with a distance metric between the antecedent and the consequent measured by the number of user clicks to go from one to another.

Two features of our model distinguish it from previous work. The first feature is the concept of distance between the antecedent and the consequent. This concept is very important for the prediction system because it allows the rules to express not only what pages are going to be accessed but also precisely when they are going to be accessed. This is especially useful for prefetching applications or for recommendation systems. Additionally, the concept of distance can be used as a measure of the rules's quality. For example if we want to redesign web pages for wireless devices by finding shortcuts, the distance of a rule can be used to measure how useful that shortcut is.

Second, our model addresses a shortcoming of current algorithms for prediction, which, traditionally, have not taken into account the characteristics of the specific web server they are trying to model. The prediction system we propose is customizable. This means that the prediction system can be adapted, depending on the characteristics of the server (number of pages, architecture of the server, number of links per page, etc.), in order to more accurately capture the behavior of its users. The model offers a balance between the storage space needed and the accuracy of the prediction system. The balance will be mainly determined by the application that is going to be developed using the prediction system.

The organization of the paper is as follows. Section 2 summarizes the motivation and prior work done in this area. Section 3 presents the Customizable Sequential Behavior Model. Section 4 implements some examples of the Customizable Sequential Behavior Model and analyzes the results. Finally, in Section 5 we conclude the article and discuss future work.

2 Motivation and Related Work

The main techniques used for pattern discovery are clustering and association rules [25].

Clustering, applied in the context of web mining, is a technique that makes it possible to group similar browsing patterns or to divide the web pages of a site into groups that are accessed together. This information can be used in the recommendation process of a page [18] or by search engines to display related pages along with their results. Also, in this context, clustering has a distinguishable characteristic: it is done with non-numerical data. This implies that, usually, the clustering techniques applied are relational. This means that we have numerical values representing the degrees to which two objects of the data set are related. Some examples of relational clustering applied to web mining are

[18] and [14]. Several authors have also considered the inherent fuzziness of the data presented in the web mining problem and have developed relational fuzzy clustering algorithms [15].

Association rule discovery aims at discovering all frequent patterns among transactions. The problem was originally introduced by Agrawal et al. [1] and is based on detecting frequent itemsets in a market basket. In the context of web usage mining, association rules refer to sets of pages that are accessed together. Usually these rules should have a minimum support and confidence to be valid. The Apriori algorithm [1] is widely accepted to solve this problem. Association rules can be used to re-structure a web site [24], to find shortcuts, an application especially useful for wireless devices [4], or to prefetch web pages to reduce the final latency [11].

The data used to obtain frequent patterns in a web mining problem has a very important characteristic: it is sequential. The user accesses a set of pages in a given order and it is very important to capture this order in the final model obtained. Unfortunately, the two previous methods lack any kind of representation of this order. Clustering identifies groups of pages that are accessed together without storing any information about the sequence. Association rules indicate groups that are presented together.

Some authors have already dealt with the problem of capturing sequentiality in association rules for web mining. The approach taken in [9] considers sequences of each session to produce rules. [21] presents the PPM algorithm, which also preserves the order of access and basically uses a Markov prediction model. The main limitation of most of these approaches is that those algorithms only detect patterns that correspond to consecutive sequences.

In this paper we present a model that is able to detect patterns produced by non consecutive sequences and that additionally preserves the order in which those web pages are visited. The model expresses those patterns using rules.

This ability to detect patterns constructed with non-consecutive sequences introduces the possibility of measuring the distance between the antecedent and the consequent of a rule. Some algorithms, like [19], are designed to detect non consecutive sequences, but there is no indication of the distance between them. In our model the distance between the antecedent and the consequent is measured in terms of the number of user clicks to go from one to the other. This concept is very important for any application (such as a recommendation system) that attempts to infer characteristics of a web site because it provides information about when the pages are going to be visited. This concept is different from finding association rules that have explicit temporal information, as done in [3], or from looking for rules that give temporal relations between different sessions of the same user, as done in [17]. Our method gives a temporal relation within the same session between the antecedent and the consequent by measuring the distance between them.

Traditionally, the models and algorithms developed for user access prediction ([2],[4],[26], etc.) apply the same approach to all servers regardless of their characteristics. In our approach, we recognize that the ability of the model to capture

user behavior depends on the characteristics of the site. In order to capture efficiently user's behavior, the model we propose is customizable. This means that it can be adapted to the inherent characteristics (number of web pages, number of users, architecture of the site, etc.) of each server. This customization capability makes possible trade-offs between the number of rules and the prediction accuracy of the prediction system.

3 Customizable Sequential Behavior Model

In this section we present the algorithms and ideas behind the concept of Customizable Sequential Behavior Model. We begin with the preparation of the data and then we present the Clustering of Users. Next, the concept of Sequential Association Rule and the definition of Customizable Behavior Model is given.

3.1 Preparing the Data

The syntax of the log file that contains all requests that a site has processed is specified in the CERN Common Log Format [7]. Basically an entry consists of (1) the user's IP address, (2) the remote logname of the user, (3) the access date and time, (4) the request method, (5) the URL of the page , (6) the protocol (HTTP 1.0, HTTP 1.1,etc.), (7) the return code and (8) the number of bytes transmitted. The W3C Web Characterization Activity (WCA) [29] defines a user session as the click-stream of page views for a single user across the entire Web. In our context, the information we really want to obtain is a server session, defined as the click-stream in a user session for a particular web server. A click-stream is defined as a sequential series of page view requests of a user. Also, the W3C defines a user as a single individual that is accessing one or more servers from a browser.

The first step for preparing the data is the transformation of the set of logs into sessions. We have defined a compiler that transforms a set of log entries L,

$$\begin{gathered} L = \left\{L_1, ..., L_{|L|}\right\} \\ L_i = (IP_i, LOGNAME_i, TIME_i, METHOD_i, URL_i, \\ PROT_i, CODE_i, BYTES_i), \forall i / i = 1...|L|, \end{gathered} \tag{1}$$

into a set of sessions S,

$$S = \left\{S_1, ..., S_{|S|}\right\}, \tag{2}$$

where $|L|$ is the number of log entries in L and $|S|$ is the number of sessions of S. Each session is defined as a tuple *(USER,PAGES)*:

$$S_i = (USER_i, PAGES_i), PAGES_i = \left\{url_{i,1}, ..., url_{i,p_i}\right\}, i = 1... \left|S\right|, \tag{3}$$

where *USER* identifies the user of the session, *PAGES* the set of pages requested, and p_i is the number of pages requested by user $USER_i$ in session S_i.

```
Input : L, Δt, |L|
Output : S, |S|
function Compiler(L, Δt, |L|)
  for each L_i of L
    if METHOD_i is GET and URL_i is WEB_PAGE then
      if ∃ S_k ∈ OPEN SESSIONS with USER_k = USER_i
        if (TIME_i − END_TIME(S_k)) < Δt then
          S_k = (USER_k, PAGES_k ∪ URL_i)
        else
          CLOSE SESSION (S_k)
          OPEN NEW SESSION(USER_i, (URL_i))
        end if
      else
        OPEN NEW SESSION(USER_i, (URL_i))
      end if
    end if
  end for
```

Fig. 1. Compiler that transforms web log data into a set of sessions.

Working with the CERN Common Log Format, a user *USER* is defined as,

$$USER_i = (IP_l, LOGNAME_{l,m}), 1 \leq l \leq d, 1 \leq m \leq h(l), \tag{4}$$

where d is the number of different *IPs* of the log, and *h(i), i=1,...,d*, the number of different *LOGNAMEs* of IP_i.

The definition of a user is highly dependent on the log format and on the information available. In order to generalize the algorithms and ideas presented we are going to work with a generic definition of user, *USER*.

The set of *URLs* that form a session satisfy the requirement that the time elapsed between two consecutive requests is smaller than Δt. The value we have used is 30 minutes, based on the results of [6] and [25]. Figure 1 presents the algorithm of the compiler. The filters implemented by our algorithm delete all entry logs that do not refer to a *URL* or that indicate an error. Also, sessions of length one or sessions three times as long as the average length of the set of sessions S are erased. This is done to eliminate the noise that random accesses or search engines would introduce to the model.

Finally, the average length of the set of sessions S is defined as,

$$N = \frac{\sum_{i=1}^{|S|} p_i}{|S|}. \tag{5}$$

3.2 Clustering of Users and Sessions

The set of different users of a system is expressed by,

$$USERS = \left\{USER_1, ..., USER_{|USERS|}\right\}, \tag{6}$$

where $|USERS|$ is the total number of different users of the log. This set of users is going to be clustered according to a cluster policy function P in order to create a customizable model. The purpose of these clusters is to group the users that have the same behavior and to allow a trade-off between the size and the personalization capabilities provided by the model.

Given p the number of clusters defined by the function P, the set of clusters of users CU can be expressed as:

$$\begin{array}{l} P : USER_i \rightarrow \{P_1, ..., P_p\}, \forall i = 1, ..., |USERS| \\ CU = \{CU_1, ..., CU_p\} \\ CU_i = \{USER_x / P(USER_x) = P_i\}, i = 1, ..., p \text{ and } 1 \leq x \leq |USERS| \\ \bigcup_{i=1}^{p} CU_i = USERS \\ CU_i \cap CU_k = \emptyset, \forall i, k \text{ with } i \neq k \end{array} \tag{7}$$

For example, given the web server of a university department a possible clustering policy function is P={*Professor, Graduate_Students, Students, Others*}.

The classification of users is going to be used to cluster the set of sessions S. The set of clustered sessions CS, can be expressed as:

$$\begin{array}{l} CS = \{CS_1, ..., CS_p\} \\ CS_i = \cup S_k / USER_k \in CU_i, \forall i = 1, ..., p, \forall k = 1, ..., |S| \end{array} \tag{8}$$

Each CS_i groups the sessions of the users that are part of the same CU_i cluster.

3.3 Sequential Association Rules

The concept of Sequential Association Rule (SAR) is based on the notion of N-Gram. In the context of web mining, an N-Gram of a session S_i is defined as any subset of N consecutive *URLs* of that session.

A Sequential Association Rule (SAR) relates an antecedent A to a consequent C in terms of the temporal distance between them. Given $|A|$ the length of the sequence of URLs of the antecedent, $|C|$ the length of the sequence of URLs of the consequent, and n the distance between the antecedent and the consequent, the SAR is defined as follows:

$$\begin{array}{c} A \underset{n}{\rightarrow} C \\ A := url_j, ..., url_{|A|+j} \\ C := url_l, ..., url_{|C|+l} \\ l = |A| + j + n + 1, \ n \in \mathrm{N}^+ \end{array} \tag{9}$$

A SAR expresses the following relation: if the last click stream of length $|A|$ of a session is A, then, in n clicks, the set of URLs C will be requested in that same session.

Each SAR is constructed from an N-Gram obtained from a session. This means that for the SAR of the definition some session S_k of a given CS_i contains an N-Gram, with N=$|A| + |C| + n$, that satisfies,

$Input : |A|, n, |C|, CS_i$
$Output : SR(CS_i)_{|A|,n,|C|}$
$function$ Obtain_SR($|A|, n, |C|, CS_i$)
  $SR(CS_i)_{|A|,n,|C|} = \emptyset$
  for each S_k ofCS_i
    for j = 1 to p_k
      if j + $|A| + n + |C| \leq p_k$
        $SAR = url_{k,j}, ..., url_{k,j+|A|} \underset{n}{\rightarrow} url_{k,j+|A|+n}, ..., url_{k,j+|A|+n+|C|}$
        if (SAR,$Counter$) $\in SR(CS_i)_{|A|,n,|C|}$ then
          $Counter = Counter + 1$
        else
          $SR(CS_i)_{|A|,n,|C|} = SR(CS_i)_{|A|,n,|C|} \cup (SAR, 1)$
        end if
      end if
    end for
  end for

Fig. 2. Pseudo-code of the algorithm developed to obtain $SR(CS_i)_{|A|,n,|C|}$.

$$S_k = (..., url_{k,j}, ..., url_{k,j+|A|}, url_{k,j+|A|+1}, ..., url_{k,j+|A|+n}, url_{k,l}, ..., url_{k,l+|C|}, ...), \text{ with } l = j + |A| + n + 1 \ . \tag{10}$$

Each SAR has a degree of support and confidence associated with it. The support of a rule is defined as the fraction of strings in the set of sessions of CS_i where the rule successfully applies. The support of a rule is given by,

$$\theta(A?_1...?_nC) = \frac{|S_k \in CS_i / (A?_1...?_nC) \in S_k|}{|CS_i|}, \tag{11}$$

where $?_1...?_n$ represents the set of any n pages in the session, and $A?_1...?_nC \in S_k$ is defined as an N-Gram of S_k. This is the way to model the distance between the antecedent and the consequent when obtaining the support. The confidence of a rule is defined as the fraction of times for which if the antecedent A is satisfied, the consequent C is also true in n clicks. The confidence of the rule is defined as,

$$\sigma(A?_1...?_nC) = \frac{\theta(A?_1...?_nC)}{\theta(A)}. \tag{12}$$

$SR(CS_i)_{|A|,n,|C|}$,for the set of sessions grouped by CS_i, is defined as a set of tuples *(SAR,Counter)*, where each tuple has a SAR with an antecedent of length $|A|$, a consequent with length $|C|$ and a distance n between the antecedent and the consequent. The *Counter* of each tuple indicates the number of times that the correspondent rule occurs in S. Figure 2 shows the algorithm used to obtain $SR(CS_i)_{|A|,n,|C|}$.

In order to preserve the relevant information, only those *SARs* which have support and confidence bigger than a given threshold are considered. With θ'

the threshold of the support and σ' the threshold of the confidence, we will talk about the set of rules $SR(CS_i)_{|A|,n,|C|,\theta',\sigma'}$ as the rules of $SR(CS_i)_{|A|,n,|C|}$with a support bigger that θ' and a confidence bigger than σ'.

$SR(CS_i)_{|A|,n,|C|,\theta',\sigma'}$is defined as a set of elements $(SAR,\theta_{SAR},\sigma_{SAR})$, where SAR is a Sequential Association Rule, and θ_{SAR} and $\sigma_{SAR}$the support and confidence associated with it, satisfying $\theta_{SAR} > \theta$' and $\sigma_{SAR} > \sigma$'. $SR(CS_i)_{|A|,n,|C|}$ contains enough information to obtain $SR(CS_i)_{|A|,n,|C|,\theta',\sigma'}$.

In order to optimize the storage and access to the rules that define the set of rules $SR(CS_i)_{|A|,n,|C|,\theta',\sigma'}$, we group the rules with the same antecedent, storing for each set of rules the consequent and the degree of support and confidence associated with it. The structure of $SR(CS_i)_{|A|,n,|C|,\theta',\sigma'}$ is:

$$SR(CS_i)_{|A|,n,|C|,\theta',\sigma'}(A) = \begin{cases} ((C_{1,1},\theta_{1,1},\sigma_{1,1}),...,(C_{1,l_1},\theta_{1,l_1},\sigma_{1,l_1})), & A = A_1 \\ ... \\ ((C_{k,1},\theta_{k,1},\sigma_{k,1}),...,(C_{k,l_k},\theta_{k,l_k},\sigma_{k,l_k})), & A = A_k \\ \emptyset, \text{ Otherwise} \end{cases} \quad (13)$$

where A is a click stream of length $|A|$, A_i, i=1...k, with $|A_i| = |A|$, is the set of antecedents of the rules of $SR(CS_i)_{|A|,n,|C|,\theta',\sigma'}$, $C_{i,j}$, i=1...k, j=1...l_k , with $|C_{i,j}| = |C|$, is the set of consequents for each antecedent A_i, and l_k is the number of consequents of each antecedent.

3.4 Sequential Behavior Model

The concept of Sequential Behavior Model (SBM) defines a prediction system based on the Sequential Association Rules introduced in the previous section. A Sequential Behavior Model is defined by a tuple $\{RU,\Phi\}$ where RU is a set of rules and Φ is the decision policy function.

RU is defined as:

$$RU(USER,A) = \begin{cases} SR(CS_1)_{|A|,n,|C|,\theta',\sigma'}(A), \text{if USER} \in CU_1 \\ ... \\ SR(CS_p)_{|A|,n,|C|,\theta',\sigma'}(A), \text{if USER} \in CU_p \end{cases} \quad (14)$$

The function Φ is defined as:

$$\Phi((C_{i,1},\theta_{i,1},\sigma_{i,1}),...,(C_{i,l_i},\theta_{i,l_i},\sigma_{i,l_i})) = \{C_{i,j},...,C_{i,b}\} \\ with\ 1 \le \text{i} \le \text{k},\ 1 \le j \le l_k, 1 \le b \le l_k, \quad (15)$$

where the independent variable of Φ is the set of consequents obtained from RU for a given user and a given antecedent (click stream), and the dependent variable is the consequent or set of consequents predicted. Some examples of policy functions include,

$$\Phi((C_{i,1},\theta_{i,1},\sigma_{i,1}),...,(C_{i,l_i},\theta_{i,l_i},\sigma_{i,l_i})) = \{C_{i,j}\} \\ with\ \sigma_{i,j} \ge \sigma_{i,m} \forall m/m = 1...l_i, m \ne j, \quad (16)$$

$$\Phi((C_{i,1},\theta_{i,1},\sigma_{i,1}),...,(C_{i,l_i},\theta_{i,l_i},\sigma_{i,l_i})) = \{C_{i,j}\} \\ with\ \theta_{i,j} \ge \theta_{i,m} \forall m/m = 1...l_i, m \ne j, \quad (17)$$

$$\begin{array}{l} \Phi((C_{i,1},\theta_{i,1},\sigma_{i,1}),...,(C_{i,l_i},\theta_{i,l_i},\sigma_{i,l_i})) = \{C_{i,j},C_{i,b}\} \\ with\ \sigma_{i,j} \geq \sigma_{i,b} \geq \sigma_{i,m} \forall m/m = 1...l_i, m \neq j, m \neq b. \end{array} \tag{18}$$

The first example predicts the consequent with the biggest confidence, the second one predicts the one with the biggest support, and the third one picks the two consequents with the highest confidence. The policy function can be defined depending on the specific application and on the characteristics of the web log used.

The tuple *(RU,Φ)* defines the on-line execution of the prediction system. Given A the click stream of the last $|A|$ pages requested by the user *USER*, the set of predicted pages will be given by,

$$\Phi(RU(USER,A)). \tag{19}$$

3.5 Classes of Sequential Behavior Models

The web servers found in Internet have very different sets of characteristics, differing in the number of users, the number of web pages that the server has, the architecture of the web server, the number of links per page, etc.

The prediction systems developed to date ([2],[4],[11],[26], etc.) do not consider the different characteristics of the web servers in order to more accurately model their behavior. We recognize that the same model cannot be used to generate a prediction system for a university department server and to capture the behavior of an e-commerce site, because the two sites exhibit vastly different behavior. In order to solve this problem, our Sequential Behavior Model is customizable. This means that it can be adapted to the characteristics of the server to more accurately capture its behavior. The Model presents three different classes: global, personalized and cluster.

Global Sequential Behavior Model (GSBM). The GSBM considers only a single cluster of users. This means that the system will also consider just one cluster of the set of sessions. Formally,

$$\begin{array}{l} p = 1 \\ CU = \{CU_1\} \\ P = \{P_1\}; P_1 = \text{"Any user of the system"} \\ CS = \{CS_1\} = S \end{array} \tag{20}$$

The Global Sequential Behavior Model will be defined by the tuple *(RU,Φ)*, where RU is expressed as:

$$RU(USER,A) = \left\{SR(CS_1)_{|A|,n,|C|,\theta',\sigma'}(A), \text{if } USER \in CU_1\ . \right. \tag{21}$$

The storage capacity needed by a GSBM is generally small because it has only one set of rules. On the other hand, the personalization capabilities offered are usually limited.

Clustered Sequential Behavior Model (CSBM). The Clustered Model considers a set of p clusters. These clusters should be designed to group the set of users that have common behavior. Each set of users will end up having a set of rules that describe their behavior in RU.

This profile produces bigger models than the Global approach but also increases the personalization capabilities. The size and the prediction accuracy will depend on the number of clusters that the system has defined.

Personalized Sequential Behavior Model (PSBM). The Personalized Model is a particular example of the Clustered Model. In this case, each one of the users of a system is placed in its own cluster, which means that each user has its own set of rules to describe his/her behavior. The storage space needed by this profile is, typically, bigger than the other two but it also offers more personalization capabilities.

This profile is very interesting considering that an increasing number of sites identify its users (using password or cookies) to provide them with personalized services. Having a model that describes the behavior of each user will make it possible to provide a higher degree of adaptation and personalization.

The main criticism that arises from this idea is that too much space is needed to store each user's set of personalized sequential association rules. Nevertheless, e-sites already posses a lot of information about each user, ranging from name, address, or credit cards numbers to layouts and preferences, as can be seen in [5],[16] and [28]. The inclusion of a personal set of association rules, as will be shown in section 4.5, will not cause a large increase in the amount of needed storage.

Additionally, it is not necessary for all the users of a server that implements a Personalized Sequential Behavior Model to have a personal set of rules. A more useful approach considers that, given the set of users of a server, a smaller subset is typically responsible for the largest part of the system's load. Taking this characteristic into account, each member of this set of frequent users will have a personal cluster. The non-frequent users will be grouped in another cluster and will share a set of rules. Formally, being g the number of frequent users in the system,

$$\begin{aligned} &p = g + 1 \\ &CU = \{CU_1, ..., CU_g, CU_{g+1}\} \\ &P = \{P_1, ..., P_g, P_{g+1}\} \\ &CS = \{CS_1, ..., CS_g, CS_{g+1}\} \end{aligned} \tag{22}$$

The number of clusters is $g+1$, where the last cluster groups the non-frequent users of the system. The clustering policy is defined as P_1=*Frequent User #1*, ...,P_g=*Frequent User #g,* P_{g+1}=*All non-frequent users.* CS_1 groups the set of sessions of the frequent user #1 and CS_{g+1} groups all the sessions of the non-frequent users.

The Personalized Sequential Behavior Model can be defined by the tuple (RU,Φ), where RU is expressed as:

$$RU(USER, A) = \begin{cases} SR(CS_1)_{|A|,n,|C|,\theta',\sigma'}(A), if \text{ USER is Frequent User 1} \\ ... \\ SR(CS_g)_{|A|,n,|C|,\theta',\sigma'}(A), if \text{ USER is Frequent User g} \\ SR(CS_{g+1})_{|A|,n,|C|,\theta',\sigma'}(A), if \text{ USER} \in \text{CU}_{g+1} \end{cases} \quad (23)$$

3.6 General Rules for Application of the Classes.

This set of classes give the designer of the prediction system the capability of choosing an appropriately trade-off between the memory needed by the set of rules and the personalization capabilities provided. Each one of the profiles is useful in different contexts:

• The Global Sequential Behavior Model is likely able to accurately capture the user behavior of a simple site. We consider a simple site as one that has a small number of users and/or a small number of web pages and/or a simple architecture (normally a tree architecture). A typical example of this kind of site is the web server of a university department.

• The Clustered Sequential Behavior Model is capable of modeling the behavior of more complex systems. A complex system is defined as a system with a high number of users and/or a high number of web pages and/or a highly interconnected structure. The definition of the clustering policy function P is application dependent.

• The Personalized Sequential Behavior Model is recommended for highly complex sites and/or for sites that need individual information for each user.

4 Examples of Sequential Behavior Models

This section evaluates the performance of different Sequential Behavior Models for different web logs.

4.1 Characteristics of the Logs Used

e have selected three representative sets of logs in order to cover different types of servers, ranging from small sites with few users to complex commercial sites with a large number of users.

The first log is from the Computer Science Department (CS) of the Polytechnic University of Madrid [10]. The training set is for the month of September 2001. The test set has been defined using log data for a single day, 1^{st} October 2001. This is an example of a small site, with a simple tree architecture and a small number of visitors.

The second log is from the NASA Kennedy Space Center server [20]. It contains 3,461,612 requests collected over the months of July and August 1995.

This is an example of a medium site server, with a complex architecture and a medium number of users. The training set considered was for the month of July 1995, and the test set has been defined using log data for the 1^{st} of August 1995.

The third site is ClarkNet [8], a commercial Internet site provider, which contains 3,328,587 requests over a period of two weeks. This is an example of a large commercial site, with a highly complex architecture and a high number of visitors. The training set has been defined from 28 August 1995 to 9 Sep 1995, and the test set is the log data for 10^{th} Sep 1995.

These training sets are the inputs to the algorithms and filters presented in Section 3. Table 1 presents some characteristics of the training sets including the processing time of the algorithm presented in Fig. 1 for each log. Although the filtering process eliminates a lot of sessions, we keep the relevant part of the requests. The processing time is given for an implementation in CLISP of the compiler, running with Linux, on a Pentium III 450MHz machine.

Table 1. Training Log Characteristics.

	CS	NASA	CLARKNET
Size	27M	160M	308.6M
Dates	Sep. 2001	Jul. 1995	28 Aug.-9 Sep. 1995
Processing Time	12 min	4 h 35 min	8 h 50 min
# of sessions before filter	3,499	124,666	224,935
# of sessions after filter	839	57,875	83,011
# of HTML requests before filtering	6,244	352,844	511,536
# of HTML requests after filtering	3,504	215,223	283,844

The characteristics of the test logs defined for each one of the training sets are given in Table 2. The processing time indicates the time needed to obtain the set of sessions from the logs using the algorithm presented in Fig. 1. The number of sessions and number of pages requested shown in Table 2 are post-filtering.

Table 2. Test Log Characteristics.

	CS	NASA	CLARKNET
Size	170K	6.8M	19M
Date	Oct. 1^{st} 2001	Aug. 1^{st} 1995	Sep. 10^{th} 1995
# of sessions	53	2753	5725
# of Pages requested	138	9521	18233
Average length of Session	2.6	3.4	3.9
Processing Time	25 sec.	2 min 40 sec	6 min 10 sec

4.2 Implementation of Global Sequential Behavior Models

We have obtained the Global Sequential Behavior Model of each log as defined by $SR(CS_1)_{1,1,1}$. After that, a threshold of 1% for the support and of 5% for the confidence has been applied, obtaining $SR(CS_1)_{1,1,1,1,5}$. Other values of support and confidence were tested, but the best prediction rate is obtained with 1% and 5%. The characteristics of these *SRs* are presented in Table 3. The processing time indicates the time needed to process each one of the training sets using the algorithm presented in Fig. 2.

Table 3. Characteristics of the SR obtained.

	CS	NASA	CLARKNET
Processing Time	50 sec.	3 min 20 sec.	7 min. 40 sec.
# of SAR of SR before applying thresholds	392	15,644	49,245
# of SAR of SR after applying thresholds	94	116	166

Fig. 3 presents the percentage of correct prediction for CS, NASA and ClarkNet logs, with RU defined by $SR(CS_1)_{1,1,1,1,5}$ and different Φ functions. The percentage of correct prediction is obtained comparing the actual next page of the session with the page or set of pages predicted by the system. The first set of columns represents the results for the function Φ defined as:

$$\begin{array}{l}\Phi_1((C_{i,1},\theta_{i,1},\sigma_{i,1}),...,(C_{i,l_i},\theta_{i,l_i},\sigma_{i,l_i})) = \{C_{i,j}\} \\ with\ \sigma_{i,j} \geq \sigma_{i,m} \forall m/m = 1...l_i, m \neq j.\end{array} \tag{24}$$

In other words, only the consequent with the highest confidence is provided as the prediction. The rest of the functions Φ defined for each set of columns are, in order:

$$\begin{array}{l}\Phi_2((C_{i,1},\theta_{i,1},\sigma_{i,1}),...,(C_{i,l_i},\theta_{i,l_i},\sigma_{i,l_i})) = \{C_{i,j},C_{i,b}\} \\ with\ \sigma_{i,j} \geq \sigma_{i,b} \geq \sigma_{i,m} \forall m/m = 1...l_i, m \neq j, m \neq b,\end{array} \tag{25}$$

$$\begin{array}{l}\Phi_3((C_{i,1},\theta_{i,1},\sigma_{i,1}),...,(C_{i,l_i},\theta_{i,l_i},\sigma_{i,l_i})) = \\ \{C_{i,j},C_{i,b},C_{i,h}\}\ with\ \sigma_{i,j} \geq \sigma_{i,b} \geq \sigma_{i,h} \geq \sigma_{i,m} \\ \forall m/m = 1...l_i, m \neq j, m \neq b, m \neq h,\end{array} \tag{26}$$

$$\Phi_4((C_{i,1},\theta_{i,1},\sigma_{i,1}),...,(C_{i,l_i},\theta_{i,l_i},\sigma_{i,l_i})) = \{C_{i,1},...,C_{i,l_i}\}. \tag{27}$$

Function Φ_2 gives as the set of predicted pages the two consequents with highest confidence, Φ_3 the set of three consequents with highest confidence, and Φ_4 all the predictions that have a support and a confidence bigger than the thresholds used.

The inability of the GSBM, as shown in Fig. 3, to efficiently capture global behavior in sites with complex structures, like NASA and ClarkNet, suggests that a more local approach should be taken.

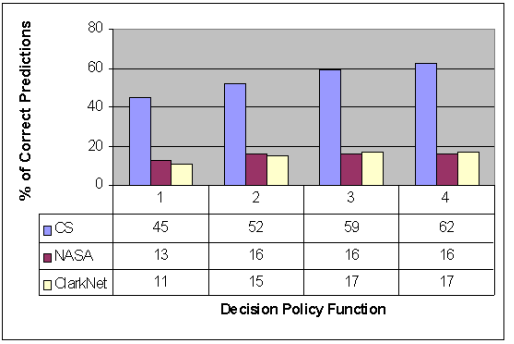


Fig. 3. Results of the model for each set of logs.

4.3 Implementation of Clustered Sequential Behavior Models Based on IP

In this section we present two CSBM examples for which the clustering policy function is defined using the IP address. In the first example, IP/16, the set of users that have the same 16 first bits of their IP equal are going to be part of the same cluster. In the second one, IP/24, the clusters will be obtained using the 24 first bits, working at the subnet level. The objective of this policy is to cluster the users that share a common behavior. This is based on the observation that the users that come from the same subnet have a greater probability of having similar behavior than two users that have completely different IPs. Table 4 presents the characteristics of the clusters for IP/16 .

Table 4. Characteristics of the clusters of IP/16.

	CS	NASA	CLARKNET
# of different IPs	840	37,821	58,035
# of Clusters	236	6,667	8,510
Processing Time to obtain Clusters	2 sec.	2 min. 10 sec.	3 min 40 sec.
Average # of IP per cluster	3.5	5.6	6.8
Processing Time of $SR(CS_i)_{1,1,1,1,5}$,*i=1,...,p*	15 sec.	40 sec.	1 min. 30 sec.
Average # of SAR per cluster	4.6	6.3	7.2

For each cluster we have obtained $SR(CS_i)_{1,1,1,1,5}$, *i=1,...,p,* with $p = 236$ for the CS log, $p = 6,667$ for the NASA log, and $p = 8,510$ for the ClarkNet log. Table 5 presents the characteristics of the clusters for IP/24.

For each cluster we have obtained $SR(CS_i)_{1,1,1,1,5}$, *i=1,...,p,* with $p = 415$ for the CS log, $p = 23,183$ for the NASA log, and $p = 36,257$ for the ClarkNet log. Fig. 4 presents the accuracy of the prediction system for IP/16 and IP/24 for each log using the decision policy functions Φ_1 and Φ_2 previously defined.

Table 5. Characteristics of the clusters of IP/24.

	CS	NASA	CLARKNET
# of different IPs	840	37,821	58,035
# of Clusters	415	23,183	36,257
Processing Time needed to obtain the Clusters	2 sec.	2 min. 40 sec.	6 min. 40 sec.
Average # of IP per cluster	2.0	1.6	1.6
Processing Time for $SR(CS_i)_{1,1,1,1,5}$,i=1,...,p	20 sec.	55 sec.	2 min. 10 sec.
Average # of SAR per cluster	2.9	6.2	5.2

In order to generate an efficient CSBM the training log should contain elements of all the possible clusters that the set of visitors of the system may form. This allows the prediction system to give a prediction for any possible visitor. If the training log lacks some of these users, the prediction system will not be able to give a recommendation for them.

In the testing logs defined for CS, NASA and ClarkNet only 2%, 5% and 6% of the sessions correspond to a user that is not included in any cluster. For those users, the system's prediction is obviously wrong. When the percentage of sessions that are not part of any of the clusters generated with the training log is significant compared with the total amount of visits, a GSBM can be applied for those users. This solution is similar to the idea presented in PSBM, where a cluster groups all non-frequent users of the system.

4.4 Implementation of Personalized Sequential Behavior Models

As presented in section 3.4.3 PSBMs are especially suitable for systems where a core of users is responsible for the best part of the load. As it will be shown, both NASA and ClarkNet logs have this characteristic. For these logs, we define a frequent user as the one that has more than two sessions in the training log. Also, because the logs lack from any kind of information about each individual

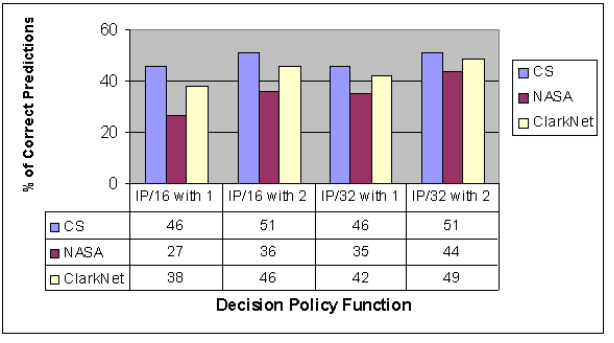


Fig. 4. Results of the CSBM with IP/16 and IP/24.

user, we will identify an IP as an user, bearing in mind that a single IP can be used by a group of users.

Table 6 shows the characteristics of our training logs, focusing this time on the data needed to identify the set of frequent users.

Table 6. Some characteristics of the Training Logs.

	NASA	CLARKNET
# of Sessions	124,666	224,935
# of Users	37,821	58,035
# of Users with just one session	30,875	49,085

As can be seen in Table 6, in the case of the NASA log, 18% of the users are responsible for 75% of the visits. Similar behavior is also observed for the ClarkNet log, where 15% of the users are responsible for 78% of the visits. These sites posseses the perfect characteristics for the implementation of a PSBM. For each frequent user we obtain $SR(CS_i)_{1,1,1,1,5}$, *i=1,...,g* where $g = 6,946$ for the NASA log, and $g = 8,950$ for ClarkNet. Table 7 presents the characteristics of the *RUs* constructed.

Table 7. Characteristics of the Personal SR constructed.

	NASA	CLARKNET
g	6946	8950
Average # of rules per $SR(CS_i)$, *i=1,...,g*	10.3	9.7
Total Processing Time	1 min 4 sec	2 min 30 sec

Fig. 5 presents the prediction accuracy of the set of PSBM constructed for the NASA log and the ClarkNet log. The test logs, in this case, have been modified to contain only the set of visits of the frequent users. This allows us to obtain the prediction accuracy of the set of personal *SRs*, $SR(CS_i)$, *i=1,...,g*.

Each set of columns of Fig. 5 presents the percentage of pages correctly predicted for each test log using two different policy functions. Using Φ_1, in both NASA and ClarkNet, the prediction system achieves at least a 44% correct prediction. With Φ_2, the correct prediction rate is over 50% for both logs.

The final prediction rate will be given by the prediction rate of $SR(CS_{g+1})$, that models the non-frequent users, and by the prediction rate of the set of personal *SRs*. In the case of the NASA logs, 75% of the visits are generated by frequent users. For the rest of the load, 25%, the prediction accuracy will be given by $SR(CS_{g+1})$. Using Φ_1 as the decision policy function gives a total prediction accuracy of 0.36 (0.75*0.44+0.25*0.13). Using Φ_2, the prediction accuracy goes

up to 0.43 (0.75*0.53+0.25*0.16). In the case of the ClarkNet log, when using $\Phi_1$the final accuracy is 0.33 and when using Φ_2 it is 0.42.

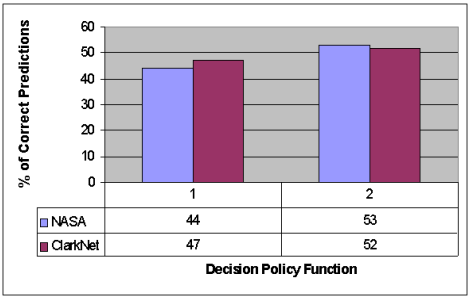


Fig. 5. Prediction Accuracy for the NASA log.

4.5 Results Analysis

As can be seen in Fig. 3, the percentage of correct prediction, using the GSBM for each log is, in the worst case, 45% for CS, 13% for NASA and 15% for ClarkNet.

The result for CS is satisfactory in the sense that the prediction system obtained would allow us to implement efficient intelligent services. Nevertheless, the results for NASA and ClarkNet are not satisfactory. The cause of this difference in the prediction system is that the sites are actually very different. In CS we have a site with a well-defined tree architecture, a low number of visitors and a low number of links per page. The other two sites have a larger number of pages, a higher number of links per page and a higher number of visitors. More importantly, they have a highly interconnected architecture. Based on these results we speculate that in those sites, user behaviors cannot be captured properly with a Global Sequential Behavior Model because access order is not a global property in complex sites.

The results of the CSBM based on IP (Fig. 4) prove that a more local approach makes possible to obtain a better prediction rate in complex sites. In this case, and using IP/16 as the clustering policy function, the NASA prediction rate, in the worst case, goes up to 27% (14% more that the GSBM) and ClarkNet up to 36% (20% more that the GSBM). Using IP/24 allows a more local approach than IP/16, and that implies an increment in the prediction rate: 35% for NASA (7% more that IP/16 and 21% more that GSBM), and 42% for ClarkNet (4% more that IP/16 and 24% more that GSBM). Nevertheless, a more local approach in the case of a simple site, like CS, does not necessarily mean a better prediction rate. As we can see, the prediction rate for CS stays basically the same for both IP/16 and IP/24. With these results we speculate that a Global Sequential Behavior Model can efficiently capture the behavior of

a site which has clear patterns, which occurs mainly in sites that have a tree architecture and a low number of visitors.

Although the CSBM increases the prediction rate of complex sites, this prediction rate is not necessarily enough for any intelligent application that is going to be implemented using the proposed model. For those cases, a PSBM can be used.

Using a PSBM, NASA prediction rate goes up to 44% (29% more that the GSBM and 17% more that the CSBM with IP/16), and ClarkNet goes up to 47% (36% more that the GSBM and 9% more that the CSBM with IP/16). And this is for the worst case, because when using more complex decision policy functions the successful prediction rate goes up to 53% for NASA and 52% for ClarkNet.

These results show that PSBM produces for highly complex sites the same successful prediction rates as the GSBM produces for simple sites. The PSBM is able to efficiently capture the behavior of complex highly interconnected sites with an acceptable increase in memory usage. This increase in memory is less significant when one takes into account that most sites already have a lot information about each user for other personalization purposes.

The results also prove that our model can be adapted to the different characteristics of the site that is going to be modeled. In other words, that the model is customizable.

5 Conclusions and Future Work

We have considered the problem of modeling web user behavior. For that purpose, a Customizable Sequential Behavior Model has been designed.

One of the distinguishable characteristics of our model is that it is customizable. The necessity of a customizable model comes from the fact that Internet servers have very different characteristics. Our model can be configured to the characteristics of the server in order to obtain the maximum possible efficiency. Our Customizable Model allows a trade-off between the number of rules and the prediction accuracy.

Also, the model is able to capture the inherent sequentiality of web visits. The model is constructed using a set of Sequential Association Rules which reflect the order of the set of URLs of the antecedent and the consequent, and also the distance between the antecedent and the consequent measured in the number n of clicks between the two click-streams. This distance makes it possible to design systems that not only predict which pages are going to be visited but also when they are going to be visited.

To the best of our knowledge, our model is the only one that is customizable and that incorporates a distance metric in its rules.

Despite the encouraging results described in Section 4, a deeper study of the impact of the parameters of the model on the prediction rate is needed. We are especially interested in studying how prediction accuracy depends on the parameter n, the distance in clicks between the antecedent and the consequent. The possibility of knowing not only what pages are going to be requested but

also when they are going to be requested can improve the efficiency of a lot of services (prefetching, recommendation systems, etc.). It will also be interesting to study the parameter C, especially study the optimum value $|C|$ for a given set of sessions. Having consequents with $|C| > 1$ is useful for some applications such as the construction of web pages in real time or prefetching.

We also plan to apply our prediction model to the NYU Home web page. NYU Home is a site that allows its users, the NYU community, to personalize their channel contents, ranging from e-mail, to news or weather forecasts. The site is slower than sites serving static content because the pages have to be generated in real-time. As can be seen, this server has the ideal characteristics for the implementation of a PSBM: it has a large set of frequent users and the system already has personal information about each user. A possible use of the prediction model we have described here is to decrease the user perceived latency at such sites by pregenerating the pages using a PSBM.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules between Sets of Items in Large Databases. In: Proceedings of the 1993 ACM SIGMOD Conference, Washington DC, USA (1993)
2. Albrecht, D., Zukerman, I., Nicholson, A.: Pre-sending documents on the WWW: A comparative study. In: IJCAI99-Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (1999)
3. Ale, J.M., Rossi, G.H.: An Approach to Discovering Temporal Association Rules. In Proceedings of SAC'00, Marh 19–21, Como, Italy, (2000) 294–300
4. Anderson, C. R., Domingos, P., Weld, D.S.: Adaptive Web Navigation for Wireless Devices. In Proceedings of the Seventeenth International Joint Conference on Artifical Intelligence (IJCAI-01) (2001)
5. Belkin, N.: Helping People Find What They Don't Know. In Communications of the ACM, Vol. 43, No. 8, (2000) 58–61
6. Catledge, L.,Pitkow, J.: Characterizing browsing behaviors on the world wide web. In Computer Networks and ISDN Systems, Vol. 27, No. 6 (1995)
7. CERN Log Format, http://www.w3.org/Daemon/User/Config/Logging.html
8. ClarkNet Internet Provider Log, http://www.web-caching.com/traces-logs.html
9. Chen, M.S., Park, J.S.: Efficient Data Mining for Path Traversal Patterns. In IEEE Transactions on Knowledge and Data Engineering, Vol. 10, No. 2 (1998) 209–221
10. Departamento de Tecnologia Fotonica Log (Univerisdad Politecnica de Madrid), http://www.dtf.fi.upm.es
11. Duchamp, D.: Prefetching Hyperlinks. In: Proc. Second USENIX Symp. on Internet Technologies and Systems, USENIX, Boulder, CO (1999) 127–138
12. Etzioni, O.: The World Wide Web: Quagmire or gold mine. In: Communications of the ACM, Volume 39, No. 11 (1996) 65–68
13. Freedman, D.: Markov Chains. In: Holden-Day Series in Probability(1971)
14. Joshi, A., Joshi, K., Krishnapuram, R.: On mining Web Access Logs. In: Proceedings of the ACM-SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, (2000) 63–69

15. Krishnapuram, R., Joshi, A., Nasraoui, O., Yi, L.: Low-Complexity Fuzzy Relational Clustering Algorithms for Web Mining. In IEEE Transactions on Fuzzy Systems, Vol. 9 (4), (2001) 595–608
16. Manber, U., Patel, A., Robinson, J.: Experience with Personaliztion on Yahoo!. In: Communications of the ACM, Vol. 43, No. 8 (2000) 35–39
17. Masseglia, F., Poncelet, P., Cicchetti, R.: An efficient algorithm for Web usage mining. In Networking and Information Systems Journal, Vol. X,n^o X (2000) 1-X
18. Mobasher, B., Cooley, R.: Automatic Personalization Based on Web Usage Mining. In Communications of the ACM, Vol 43:8 (2000)142-151
19. Nanopoulos, A., Katsaros, D., Manolopoulos, Y.: Effective Prediction of Web-user Accesses: A Data Mining Approach. In: Proceeding of the WEBKDD 2001 Workshop, San Francisco, CA (2001)
20. NASA Kennedy Space Center Log, http://www.web-caching.com/traces-logs.html
21. Palpanas, T., Mendelzon, A.: Web Prefetching using Partial Match Prediction. In Proceedings of the 4^{th} Web Catching Workshop (1999)
22. Shi, W., Wright, R., Collins, E., Karamcheti, V.: Workload Characterization of a Personalized Web Site and Its Implications for Dynamic Content Caching. In: Proceedings of the 7^{th} International Conference on Web Content Caching and Distribution (WCW'02), Boulder, CO (2002)
23. Spiliopoulou, M., Pohle, C., Faulstich, L.: Improving the Effectiveness of a Web Site with Web Usage Mining. In: Proceedings of WEBKDD99 (1999) 142–162
24. Srikant, R., Yang, Y.: Mining Web Logs to Improve Website Organization. In: Proceedings of WWW10, Hong Kong (2001) 430–437
25. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.: Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. In SIGKDD Explorations, ACM SIGKDD (2000)
26. Su, Z., Yang, Q., Zhang, H.: A prediction system for multimedia pre-fetching on the Internet. In: Proceedings of the ACM Multimedia Conference 2000, ACM (2000)
27. VanderMeer, D., Dutta, K., Datta, A.: Enabling Scalable Online Personalization on the Web. In Proceedings of EC'00 (2000) 185–196
28. Wells, N., Wolfers, J.: Finance with a Personalized Touch. In: Communications of the ACM, Vol. 43:8 (2000) 31–34
29. WWW committee web usage characterization activity. http://www.w3.org/WCA
30. Yang, Q., Tian-Yi, I., Zhang, H.: Minin High-Queality Cases for Hypertext Prediction and Prefetching. In D.W. Aha and I. Watson (Eds.): ICCBR 2002, LNAI 2080, Springer-Verlag Berlin Heidelberg (2001) 744–755
31. Yang, Q., Zhang, H., Li, I., Lu, Y.: Mining Web Logs to Improve Web Caching and Prefetching. In: N. Zhong et al (Eds.): WI 2001, LNAI 2198, Springer-Verlag Berlin Heindelberg (2001) 483–492

Coping with Sparsity in a Recommender System

André Bergholz

Xerox Research Centre Europe (Grenoble)
6 chemin de Maupertuis
38240 Meylan, France
Andre.Bergholz@xrce.xerox.com

Abstract. In this paper we report experiments that we conducted using an implementation of a recommender system called "Knowledge Pump" (KP) developed at Xerox. We repeat well-known methods such as the Pearson method, but also address common problems of recommender systems, in particular the sparsity problem. The sparsity problem is the problem of having too few ratings and hence too few correlations between users. We address this problem in two different manners. First, we introduce "transitive correlations", a mechanism to increase the number of correlations between existing users. Second, we add "agents", artificial users that rate in accordance with some predefined preferences. We show that both ideas pay off, albeit in different ways: Transitive correlations provide a small help for virtually no price, whereas rating agents improve the coverage of the system significantly but also have a negative impact on the system performance.

1 Introduction

Recommender systems have enjoyed great popularity during recent years and have been recognized as an important tool for information filtering. They help users to get personalized recommendations in a some domain. Typical domains include books, movies, and music. The strength of recommender systems comes from the fact that they don't need to analyze the content of the items they recommend. For users it is also possible to discover new items purely on the basis of positive ratings of other users. Hence, recommender systems have been classified as social filtering (as opposed to cognitive and economic filtering [10]).

In the simplest case, recommender systems, also referred to as collaborative filtering systems, consist of a database with three tables: users, items, and ratings. This database is used to find users whose opinions are similar and make use of that knowledge to compute rating predictions for items that some users have rated and others have not.

Recommender systems do suffer from some problems. The most obvious one is the *sparsity problem*. If there are only a few ratings for every item then it is impossible to compute the correlation (the degree of similarity) between users. For example, at the time of writing the well-known and popular movie "Star Wars"

O.R. Zaïane et al. (Eds.): WEBKDD 2002, LNAI 2703, pp. 86–99, 2003.

had ten reviews from the Top 1000 reviewers at Amazon.com. Consequently, this movie contributes to only 45 of the 499500 correlations among these users (less than 0.01%). A related problem is the *early-rater problem*. Even if the system is not sparse in general, any new item has initially only very few ratings. Predictions on that item will be rather inaccurate. More extreme, the system is of no use to the very first rater. Similarly, any new user needs to rate items first before the system becomes helpful to him/her. At Amazon.com another problem can be observed: the *lack-of-negative-ratings problem*. People like to review items they like, they don't bother with the rest. But that gives an incomplete picture to the system. If every item is only rated 4 or 5 by all the users, then the detection of like-minded users becomes nearly impossible.

At Xerox, we have implemented a recommender system called the "Knowledge Pump" (KP)[1] [5]. The goal of the system is to give users the possibility of sharing information by providing access to documents they might be interested in. These documents are usually Websites or files in popular document formats such as PDF and Word. In any case, documents are identified by their URL. Users of KP are organized in communities; the communities are predefined. Every user can be a member of as many communities as he/she wants. In a sense, every community acts as its own recommender system. Documents can be recommended to an arbitrary set of communities. Users can rate documents and write summaries and reviews.

In this paper we summarize experiments that we did with our system. In particular, we try to address the sparsity problem. This problem is very real in our environment and we imagine it to exist in similar systems as well. This is due to the fact that in an information-sharing environment like KP there exist many documents, but only few of them are rated by more than two or three people. We try to attack this problem by "guessing" more user correlations and by introducing artificial ratings. After pointing to related work (Section 2) we examine well-known prediction methods for recommender systems in Section 3. In Section 4 we introduce transitive correlations, which help to generate more user correlations, thereby increasing the coverage (i.e., the number of rating predictions that can be generated) of the system. Section 5 introduces rating agents, which automatically provide ratings for each document based on some metainformation. Again, this is done to increase coverage. We point out the pros and cons of such an approach and conclude with a discussion in Section 6.

2 Related Work

GroupLens is the pioneering system in the field of recommender systems [13]. It helped users to find articles in newsgroups that they might be interested in. This system introduced the idea of collaborative filtering using a neighborhood-based algorithm. Another early system is Ringo, a personalized music recommendation system [15]. It introduced the idea that is underlying the P25-method in this paper. It also introduced the idea of a threshold for the correlations. The higher the

[1] For more information about KP, please contact kprequest@xrce.xerox.com.

threshold was the better the prediction turned out to be, but the less predictions could be made at all. The Bellcore Video Recommender uses random samples of neighbors instead of the full set [8]. Fab integrates content and collaborative filtering by using content profiles to identify user tastes [3]. Other recommender systems include PHOAKS [16] and Referral Web [9].

The sparsity problem has been extensively studied in recent work. A popular approach is the content-based approach, in which the content of the documents in question is analyzed to reason about user preferences. In [14], much like in parts of our work, filtering agents based on simple observations such as document length or the number of spelling errors are introduced and integrated into the collaborative filtering framework. [6] is an extension of that paper, where more advanced agents based on user profiles and machine learning techniques are introduced. The conclusion there is that a single agent can outperform collaborative filtering, but ultimately the combination of user's opinions and agents works best. In [12] a pseudo-rating for every user-document pair is computed based on the textual categorization of the documents that were rated by the given user. Instead of using commonly rated documents the work in [11] relies on a content-based user-feature vector as the basis for correlation between users. The impact of implicit ratings, derived from the time spent on a document or the amount of scrolling, is investigated in [4]. Finally, in [2] a new method for collaborative filtering is presented, in which a directed graph is constructed where the edges denote predictability between users; paths of that graph can be used to compute predictions.

3 Basic Prediction Formulas

In this section we compare previously known techniques for the KP system. In general, the prediction of a rating for a user-document pair consists of two steps. First, correlations between the current user and other users need to be determined. Second, the ratings of the correlated users (the neighborhood) for the current document need to be transformed into a prediction for the current user. Most of the methods used here were previously described in [7].

Method Average (AV). The prediction is computed by using the average rating of that document in the current community (excluding the current user's rating if it exists).

Method Pearson (P). This method is used in many recommender systems. First, correlations between users are computed. This is done using the Pearson r correlation formula:

$$r(i_1, i_2) = \frac{\sum_k (r_{i_1 k} - \overline{r_{i_1}}) \sum_k (r_{i_2 k} - \overline{r_{i_2}})}{\sqrt{\sum_k (r_{i_1 k} - \overline{r_{i_1}})^2} \sqrt{\sum_k (r_{i_2 k} - \overline{r_{i_2}})^2}} \tag{3.1}$$

The sum is built over indices $k \in \{j | d_j \in D, \exists r_{i_1,j}, r_{i_2,j}\}$ for documents that both users have rated. The $\overline{r_i}$ indicates the average rating of the user i. The Pearson r

correlation is 1 if the two users are positively correlated, −1 if they are negatively correlated, and 0 if they are uncorrelated. The correlation is used to compute the predictions. This is done by estimating the difference of the users' document rating from the average users' ratings, weighted by the correlation between the users and the current user.

$$p_{ij} = \overline{r_i} + \frac{\sum_k (r_{kj} - \overline{r_k}) r(i,k)}{\sum_k |r(i,k)|} \tag{3.2}$$

Here, the sum is built over indices k of users that are chosen to be "correlated enough" to the current user. Possible constraints include $|r(i,k)| \geq 0.5$ or $r(i,k) \geq 0.5$. However, due to the limited number of ratings we have in KP we do not enforce such constraints.

Method Pearson 2.5 (P25). Here we replace every average $\overline{r_i}$, $\overline{r_{i_j}}$, and $\overline{r_k}$ in the Pearson formulas 3.1 and 3.2 for both correlation and prediction by the average possible rating 2.5.

Method Pearson Average (PAV). We adapted the previous method P25 and used the real overall average score from the review table instead of 2.5. This average score is 3.5594 in our dataset.

Method Pearson + Significance Weighting (PSW). This method is a variation of the Pearson method. The idea is that user correlations based on a high number of commonly rated documents should be more important than the ones based on a lower number. In our case, four or more commonly rated documents count as significance 1, three as 0.75, two as 0.5, and one as 0.25. The prediction formula is adapted as follows:

$$p_{ij} = \overline{r_i} + \frac{\sum_k (r_{kj} - \overline{r_k}) s_{ik} r(i,k)}{\sum_k |s_{ik} r(i,k)|} \tag{3.3}$$

Here, s_{ik} denotes the significance of the correlation between users i and k.

Method Pearson + Z-Scores (PZS). The idea here is to incorporate the differences in the width of a user's rating distributions. Instead of the actual ratings we use the z-scores in the formula. The z-score answers the question "How many standard deviations is the rating away from the average?". The prediction formula for this method is the following:

$$p_{ij} = \overline{r_i} + \sigma_i \frac{\sum_k \frac{r_{kj} - \overline{r_k}}{\sigma_k} r(i,k)}{\sum_k |r(i,k)|} \tag{3.4}$$

Here, the σ_i and σ_k denote the standard deviation of the rating distributions of the users i and k.

For evaluating the system we use the following measures. We measure both the accuracy of the prediction and the coverage, i.e., the number of user-document pairs for which a prediction could be made. Let n denote the number of user-document pairs where both a prediction and a rating exists.

Mean Absolute Error (MAE). We sum the absolute values of the differences between the predictions and the ratings and divide it by n. This measure is the most commonly used one.

Receiver Operating Characteristic Curve (ROC). From signal processing comes one of the most important measures in decision support. ROC sensitivity indicates how many "good" items are actually predicted to be "good"; ROC specificity indicates how many "bad" items are actually predicted to be "bad". The ROC curve plots sensitivity against 1-specificity for different cut points (i.e., points, which separate "good" and "bad"). We fix the cut point and say that for an item to be "good" it has to have a rating of 4 or 5. As the error measure, we combine sensitivity and specificity into one measure. We divide the number of user-document pairs that have a prediction of above 3.5 and a rating of 4 or 5 or a prediction of 3.5 or less and a rating of 3 or less by n. This measure represents the percentage of the correctly classified documents.

Coverage. Coverage measures for how many user-document pairs a rating prediction could be produced. We divide n by the number of user-document pairs for which we have a rating.

Table 1. Basic methods for recommender systems (Overview)

Method	MAE	ROC	Coverage
AV	0.7742	0.6369	0.5927
P	0.5292	0.8149	0.7146
P25	0.6179	0.7004	0.8587
PAV	0.5169	0.8896	0.8587
PSW	0.5365	0.8063	0.7146
PZS	0.4916	0.8301	0.7146

Table 1 shows the first results we obtained, averaged over all the communities of KP. As expected, the correlation-based ("personalized") methods perform better than the generic AV-method. Contrary to the claims made by the developers of Ringo [15] the method P25 does not perform better than the regular Pearson method. However, our new method PAV, which is using the real average of the dataset, performed surprisingly well and better than the normal Pearson method. Note that the coverage of P25 and PAV is better than the coverage of the other Pearson-based methods: More correlations can be computed, because no information about average user ratings is needed. Significance Weighting did surprisingly not lead to an improvement (contrary to the results in [7]). A reason for that may be found in the low number of ratings in our system; one may argue that four commonly rated documents still do not make a correlation very significant (or twice as significant as two commonly rated documents for that matter). The use of z-scores on the other hand seems to improve the performance (in line with the results in [7]). As a reason we imagine that z-scores take the variation in every user's ratings into better account.

4 Transitive Correlations

In this and the next section we try to address the sparsity problem. Remember that due to low rating activity it can be difficult to compute a sufficiently large set of correlations between users. In this section we evaluate the effectiveness of "transitive correlations". We obtain the correlation between users i and j by operating on the correlations between users i and k and users k and j. The users i and j may thus not have a common rating activity, yet a correlation between them can be given. The hope here is that the coverage of the system can be improved without the performance being hurt too much.

Arithmetic Mean (ARI). The first idea for computing a transitive correlation between users i and j is to use the arithmetic mean of the correlation between i and some user k and the correlation between k and j. The overall arithmetic mean over all available users k will be used. In the formula, n denotes the total number of these available users k.

$$r_t(i,j) = \frac{1}{n} \sum_k \frac{r(i,k) + r(k,j)}{2} \tag{4.1}$$

Product (PRO). A second idea is to use the product of $r(i,k)$ and $r(k,j)$. The rational behind that can be deducted from Table 2. If both "incoming" correlations are positive, the transitive correlation should be positive. If one of the "incoming" ones is negative, the other one positive, the transitive correlation should be negative, because the two positively correlated users can be seen as like-minded. The last line of the table is a little more controversial, but intuitively it is more likely that if both "incoming" correlations are negative the transitive correlation should be positive. After all, this reasoning is also applied when negative correlations are used in the prediction process.

Table 2. Assumed positive and negative transitive correlation

$r(i,k)$	$r(k,j)$	$r_t(i,j)$
+	+	+
+	−	−
−	+	−
−	−	+

$$r_t(i,j) = \frac{1}{n} \sum_k r(i,k) r(k,j) \tag{4.2}$$

Geometric Mean (GEO). The product may produce transitive correlations that are too close to zero. Hence, in this method we lessen that effect by using the square root.

$$r_t(i,j) = \frac{1}{n}\sum_k \begin{cases} \sqrt{r(i,k)r(k,j)}, & \text{if } r(i,k)r(k,j) \geq 0 \\ -\sqrt{-r(i,k)r(k,j)}, & \text{if } r(i,k)r(k,j) < 0 \end{cases} \tag{4.3}$$

Hybrid methods (HY1 and HY2). Finally, we consider two hybrids of the above methods. As a base, we use the arithmetic mean, but we try to put it more into accordance with the intuition of Table 2. That is, we use product and geometric mean where one "incoming" correlation is positive and the other one negative, and we use the negated arithmetic mean when both "incoming" correlations are negative.

$$r_t(i,j) = \frac{1}{n}\sum_k \begin{cases} \frac{r(i,k)+r(k,j)}{2}, & \text{if } r(i,k) \geq 0,\ r(k,j) \geq 0 \\ -\frac{r(i,k)+r(k,j)}{2}, & \text{if } r(i,k) < 0,\ r(k,j) < 0 \\ r(i,k)r(k,j), & \text{otherwise} \end{cases} \tag{4.4}$$

$$r_t(i,j) = \frac{1}{n}\sum_k \begin{cases} \frac{r(i,k)+r(k,j)}{2}, & \text{if } r(i,k) \geq 0,\ r(k,j) \geq 0 \\ -\frac{r(i,k)+r(k,j)}{2}, & \text{if } r(i,k) < 0,\ r(k,j) < 0 \\ -\sqrt{-r(i,k)r(k,j)}, & \text{otherwise} \end{cases} \tag{4.5}$$

We measure the error in two different manners. First, we compare the computed transitive correlations with the real correlations computed by the Pearson method (correlations-based error). Second, we evaluate Pearson prediction based on the transitive correlations (prediction-based error).

Table 3. Error of transitive correlation with respect to the real correlations

Method	MAE
ARI	0.2595
PRO	0.3140
GEO	0.2655
HY1	0.2509
HY2	0.2589

Table 3 shows the MAE of the transitive correlations based on the available real correlations computed in a Pearson-like manner. An MAE of 0.25 thus means that the transitive correlation is on average 0.25 higher or lower than the real one. We can see that except for the PRO-method all methods are of about the same quality. That comes as a little surprise as we expected GEO to perform best. Another surprising detail is that method HY1 performs very well even though it makes use of the badly performing method PRO.

Table 4 shows how prediction based on transitive correlations performs. As a means of comparison we list again the results for the Pearson method from

Table 4. Error of transitive correlation with respect to the generated predictions

Method	MAE	ROC	Coverage
Pearson	0.5292	0.8149	0.7146
ARI	0.7081	0.6811	0.6579
PRO	0.7055	0.6813	0.6579
GEO	0.7048	0.6816	0.6579
HY1	0.7036	0.6860	0.6579
HY2	0.7047	0.6850	0.6579
Real + ARI	0.5308	0.8153	0.7356
Real + PRO	0.5295	0.8169	0.7356
Real + GEO	0.5301	0.8165	0.7356
Real + HY1	0.5303	0.8165	0.7356
Real + HY2	0.5303	0.8164	0.7356

Section 3 in the first line. We did two sets of experiments. First, we used only the transitive correlations for the prediction. Second, we used the real correlations and enriched them by adding transitive correlations for those pairs of users where we do not have a real correlation. Again, we can see that all five methods perform virtually identical. We furthermore see that the results of the first set are bad, the error is much bigger than for the Pearson method and even the coverage is lower. Hence, basing the prediction solely on transitive correlations is not a good idea. For the second set, two things are surprising. The coverage is not improved by as much as we thought it would be. On the other hand, the performance is virtually not harmed at all, in fact the ROC values are even slightly better than for the Pearson method. Furthermore, it is surprising to see that method PRO, which performed worst in Table 3 is now the best performing method. We would like to add another observation we made: Every user for which we could compute correlations (76 out of 410) has an average of 8.67 correlated users per community (the standard deviation being 5.92). For the transitive correlations this number increases to 21.35 (the standard deviation being 10.20). Because we could recreate virtually every real correlation with the transitive correlations we can say that we get an average of almost 13 additional (transitive) correlations per user and community, an increase of almost 150%.

The above results are rather inconclusive. So we decided to perform similar experiments using a different dataset, the EachMovie dataset [1]. This dataset is very large, so we reduced it in the following manner. We only considered users between the age of 16 and 18 who voted at least 50 times between July 1996 and June 1997 for movies that have a theatrical release date. The resulting dataset is not sparse at all. It consists of 96 users, 630 movies, and 7287 ratings. We were able to calculate real correlations for all pairs of users. For our experiments that means two things: First, the first error measure is much more meaningful as we can compare a virtually complete set of transitive correlations with a complete set of real correlations. Second, for the second error measure (the prediction quality) it does not make much sense to use a combination of real

and transitive correlations, because this combination would be dominated by the real correlations.

Table 5. Correlation-based error for the EachMovie dataset

Method	MAE
ARI	0.1714
PRO	0.2005
GEO	0.1628
HY1	0.1601
HY2	0.1582

Table 6. Prediction-based error for the EachMovie dataset

Method	MAE	ROC	Coverage
Pearson	0.1615	0.7327	0.9918
ARI	0.1965	0.6664	0.9918
PRO	0.1856	0.6860	0.9918
GEO	0.1850	0.6894	0.9918
HY1	0.1908	0.6750	0.9918
HY2	0.1866	0.6845	0.9918

Table 5 and Table 6 summarize the results. In general, we can say that the results are much better than the results for our Knowledge Pump dataset. The transitive correlations are closer to the real ones, and the predictions (which are solely based on the transitive correlations is this case) are closer to the Pearson-based prediction as well. As for the methods, it finally seems that our expected winner, the GEO-method, plus the hybrid method HY2, which is based on GEO, perform slightly better than the rest.

We performed one final set of experiments, in which we restricted the real correlations that are used to compute the transitive correlations. We used only those correlations that are either greater or equal to 0.5 or less or equal to -0.5. In other words, we used only correlations that are either clearly positive or clearly negative and omitted the "grey zone" of correlations between -0.5 and 0.5. The Ringo system uses the same restriction idea for the normal Pearson-based prediction [15].

Tables 7 and 8 summarize the results we obtained. They are a little worse than before. The method PRO performs best in Table 7. We believe that the effect that the product produces transitive correlations close to zero is lessened by the fact that only "large" correlations of an absolute value of 0.5 or more are used as input. Unfortunately, the prediction-based error of all methods is again virtually identical. So the question for the best method must remain open. As a final remark, we'd like to add that for the basic methods from Section 3 method

Table 7. Correlation-based error for the EachMovie dataset (reduced base correlations)

Method	MAE
ARI	0.2711
PRO	0.1866
GEO	0.3136
HY1	0.2855
HY2	0.3152

Table 8. Prediction-based error for the EachMovie dataset (reduced base correlations)

Method	MAE	ROC	Coverage
Pearson	0.1615	0.7327	0.9918
ARI	0.1940	0.6849	0.8641
PRO	0.1954	0.6837	0.8641
GEO	0.1952	0.6841	0.8641
HY1	0.1941	0.6873	0.8641
HY2	0.1952	0.6794	0.8641

Pearson + Z-Scores (PZS) emerged as the winner for the EachMovie dataset as well.

In conclusion, using transitive correlations is a good idea, because it gives a (small) increase of coverage without harming the performance of the system. However, from this study it remains open which of the suggested methods to use.

5 Agents

In this section we want to further address the sparsity problem. In KP, we are faced with the problem of having too few rates. Hence, we can compute only a limited number of correlations between users. At the same time, there is plenty of metainformation on the documents in KP available, and we want to make use of it. The idea of having artificial users who rate documents based on some criterion is not new and has been tested e.g. in [14] and the follow-up paper [6]. A key advantage of this approach is that it fits nicely into the recommendation framework: both real and artificial users rate items and then correlation between pairs of users are computed no matter whether they are real or artificial. In fact, the work in [6] suggests that an effective mechanism for producing high-quality recommendations is to throw in any available data and allow the collaborative filtering engine to sort out which information is useful to each user. If one instead incorporates metainformation directly (e.g., by emphasizing new documents over old ones), one creates an ad-hoc solution which depends on the chosen parameters. Furthermore, negative correlation (some users may prefer older documents) is ignored.

Artificial users that rate have another advantage: they help in attacking the early-rater problem as well. Every new document has already ratings from the

artificial users when it is presented to the real users. Hence, some personalized recommendation is possible based on the correlations of the first real users to the artificial users. On the other hand it remains true that any new completely user needs to rate items first before the system becomes useful.

Table 9 lists the artificial users (which we call agents for reason of simplicity) we have implemented so far. They belong to two categories: KP-related (type 100, 100-199) and document-format-related (type 200, 200-299). The types of agents we use here are very generic, they should be applicable to a wide range of recommender systems in a Web environment and can easily be implemented. We think of more categories for the future: time-related (creation date, last modification date), type/content-related (home page, call for papers, minutes, etc.), and presentation-related (language, length, number of frames, number of links, number of spelling errors, etc.).

Table 9. Implemented Agents

Number	Description
101	High number of ratings
102	High number of visits
111	Recent first rating
112	Recent latest rating
113	Recent first visit
114	Recent latest visit
201	HTML
202	PDF
203	PS
204	Word

The KP-related agents are implemented on an interval basis: A certain rating is given for a certain interval of rating numbers, a certain time interval of first ratings, etc. For instance, agent 111 rates all documents whose first rating is after 01/01/2001 as five, all documents whose first rating is between 07/01/2000 and 12/31/2000 as four and so on. The document-format-related agents simply rate a document as five if it is of their preferred format and two otherwise.

Table 10. Results for the agents experiments

Method	MAE	ROC	Coverage
Pearson	0.5292	0.8149	0.7146
100-Agents	0.6014	0.7874	0.8756
200-Agents	0.6014	0.7720	0.8756
100- and 200-Agents	0.5622	0.7976	0.8756

Table 10 lists the results we obtained. We used the Pearson method and the error metrics introduced in Section 3. The first line gives the result from

the aforementioned section where no agents are used at all. Then we give the results where only type 100- or only type 200-agents are used and finally the result where all the agents are used. One thing we can observe immediately is that the coverage of the system is greatly increased once we use any kind of agent. But using the agents has its price. With only one type of agents used the prediction error does increase significantly. However, the use of all agents reduces that effect. This is not surprising, because the presence of more agents gives the system a higher chance of finding agents which are well-correlated to the current user. We would like to mention though that the presence of agents dramatically increases the number of ratings in the system, because they rate every single document, whereas real users typically rate less than 1% of the documents. Hence, the performance of the system is significantly affected. For a commercial implementation incremental maintenance of the correlations and predictions will be and sampling of neighborhood users may be necessary.

6 Discussion and Conclusion

In this paper we have described some experiments we did with our implementation of a recommender system called "Knowledge Pump" (KP). With these experiments we try to address the problems in today's recommender systems, in particular the sparsity problem. This problem arises if there are only a few ratings in the system. Then it becomes very hard to compute the correlation (the degree of similarity) between users. In Section 3 we compared basic, well-known techniques for recommender systems. We concluded that the usage of z-scores instead of actual ratings as well as the use of the overall average instead of the personalized averages may improve the performance of the standard Pearson method. Using the overall average also increases the coverage of the system and improves the performance, because computation is easier. We attack the sparsity problem from two directions. By introducing transitive correlations we try to increase the number of correlations between users and thus the coverage of the system. By introducing agents (artificial users) we try to increase the number of ratings and active users in the system and thus give the system more options to find correlated users for a given current user. Both ideas can be recommended. Though transitive correlations do not increase the coverage by as much as we expected they do not harm the performance of the system at all. Insofar they can be regarded as a small help for no price. Agents on the other hand greatly increase the coverage, but they also do harm the prediction quality. The latter effect can be reduced by using a higher number of agents. Agents do have a significant impact on the performance, though, and incremental computations need to be implemented for commercial systems. In the future, we want to gain more experience by using different datasets and investigate, how we can identify expert users in a recommender system.

Acknowledgment.
I would like to thank Boris Chidlovskii, Stefania Castellani, and Damien Prevosto for many helpful comments and discussions.

References

1. EachMovie collaborative filtering data set, http://research.compaq.com/SRC/eachmovie/.
2. C. C. Aggarwal, J. L. Wolf, K. L. Wu, and P. S. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 201–212, San Diego, CA, USA, August 1999.
3. M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.
4. M. Claypool, D. Brown, P. Le, and M. Waseda. Inferring user interest. *IEEE Internet Computing*, 5(6):32–39, 2001.
5. N. Glance, D. Arregui, and M. Dardenne. Knowledge Pump: Supporting the flow and use of knowledge. In U. Borghoff and R. Pareschi, editors, *Information Technology for Knowledge Management*, pages 35–51. Springer-Verlag, Berlin – Heidelberg – New York, 1998.
6. N. Good, B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 439–446, Orlando, FL, USA, July 1999.
7. J. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 230–237, Berkeley, CA, USA, August 1999.
8. W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, pages 194–201, Denver, CO, USA, May 1995.
9. H. Kautz, B. Selman, and M. Shah. Referral Web: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.
10. T. W. Malone, K. R. Grant, F. A. Turbak, S. A. Brobst, and M. D. Cohen. Intelligent information sharing systems. *Communications of the ACM*, 30(5):390–402, 1987.
11. S. Maneeroj, H. Kanai, and K. Hakozaki. Combining dynamic agents and collaborative filtering without sparsity rating problem for better recommendation quality. In *Proceedings of the DELOS Network of Excellence Workshop on Personalisation and Recommender Systems in Digital Libraries*, Dublin, Ireland, June 2001.
12. P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems*, New Orleans, LA, USA, September 2001.
13. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 175–186, Chapel Hill, NC, USA, October 1994.

14. B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl. Using filtering agents to improve prediction quality in the GroupLens research collaborative filtering system. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 345–354, Seattle, WA, USA, November 1998.
15. U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, pages 210–217, Denver, CO, USA, May 1995.
16. L. Terveen, Hill W., B. Amento, D. McDonald, and J. Creter. PHOAKS: A system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, 1997.

A The Dataset

For the experiments, we used the KP dataset of November, 8th 2001. At that time, we observed the following facts about the dataset:

- There are 410 users, organized in 63 communities.
- Two users are members of 53 communities, 13 users are members of just one community. 147 users are member of no community.
- The community "Knowledge Management" has 170 members, the community "Imaging" has just one member. Seven communities have no members at all.
- There are 9935 ratings in the set. The earliest is from July, 11th 1997, the latest from November, 8th 2001.
- The most active user made 1018 ratings, eleven users contributed one rating. 281 users made no ratings at all.
- The community "TechWatch" has 881 ratings, the community "Imaging" has only two. Eight communities have no ratings at all.

On the Use of Constrained Associations for Web Log Mining*

Hui Yang and Srinivasan Parthasarathy

Computer and Information Science
Ohio State University,
Columbus, OH 43235
srini@cis.ohio-state.edu

Abstract. In recent years there has been an increasing interest and a growing body of work in web usage mining as an underlying approach to capture and model the behavior of users on the web for business intelligence and browser performance enhancements. Web usage mining strategies range from strategies such as clustering and collaborative filtering, to accurately modeling sequential pattern navigation. However, many of these approaches suffer problems in terms of scalability and performance (especially online performance) due to the size and sparse nature of the data involved and the fact that many of the methods generate complex models that are less than amenable to an on-line decision making environment. In this paper, we present a new approach for mining web logs. Our approach discovers association rules that are constrained (and ordered) temporally. The approach relies on the simple premise that pages accessed recently have a greater influence on pages that will be accessed in the near future. The approach not only results in better predictions, it also prunes the rule-space significantly, thus enabling faster online prediction. Further refinements based on sequential dominance are also evaluated, and prove to be quite effective. Detailed experimental evaluation shows how the approach is quite effective in capturing a web user's access patterns; consequently, our prediction model not only has good prediction accuracy, but also is more efficient in terms of space and time complexity. The approach is also likely to generalize for e-commerce recommendation systems.

1 Introduction

In the Internet age, organizations with frequently accessed web sites, often generate and collect large volumes of data on their daily operations. Most of this information is generated automatically and collected on a day by day basis. Mining such customer accesses and usage patterns on the web is an important task that can give organizations a business edge. For instance, analyzing such information can enable organizations: to improve the Internet experience for customers via personalization; to cross-market and do targeted advertising to customer clusters in order to enhance the effectiveness of promotional campaigns; and to provide valuable information on re-structuring decisions.

Encouraged by the fact that strong regularities do exist in Web surfing patterns [9,16], researchers have been evaluating different efforts to discover these regularities and put

* This work was partially supported by an Ameritech Faculty Fellowship and NSF-DBI-9981990.

O.R. Zaïane et al. (Eds.): WEBKDD 2002, LNAI 2703, pp. 100–118, 2003.

them to good use. Given a partially conducted transaction the goal might be to predict which web pages (or items in an e-commerce transaction) will be accessed (bought) in the near future. Such predictions can be used in various ways. For example, the predictions can be tailored for recommendation and web page personalization purposes. Alternatively, the same predictions can also be used for prefetching and caching (in proxy servers) in order to efficiently access web pages.

There are two steps in this process. The first step is to extract usage patterns. The second step is to build a predictive model based on the extracted usage patterns. For the first step we rely on a variant of a popular data mining technique, i.e., association rule mining[4,5]. Specifically we extract association patterns that are constrained temporally. We formally define this notion and distinguish it from sequential patterns[6] in the next section. The constraints we impose attempt to capture the key intuition that more recently accessed web pages are likely to have a greater influence on web pages that are likely to be accessed soon. For the second step we use a rule based predictive model[11], wherein rules are ordered based on their sequential dominance and confidence. The basic idea is that once the ordering is performed we use the top few rules to predict the set of web pages that will be accessed in the near future.

Detailed experimental results on one publicly available web log dataset are presented in this paper. These experimental results indicate that the proposed approach can yield good predictability (around 70% for several cases) while restricting the size of the model significantly, thereby enabling faster online prediction. Essentially the temporally constrained association rules capture most of the useful information thereby enabling the pruning of a large number of rules (the strawman approach is to use the entire set of association rules). Similar results have been obtained for other locally available datasets.

The rest of this paper is organized as follows. Section 2 briefly reviews literature related to web usage mining. Section 3 introduces our notion of constrained association patterns and the notion of sequential dominance and how these two techniques can be used to build a predictive model for web log mining. Section 4 provides detailed experimental results. Finally, we present our conclusions and ongoing work in Section 5.

2 Background and Overview of Proposed Work

Mining web log or e-commerce data is driven by the need to derive actionable knowledge from such data efficiently. The challenges are daunting. First, the datasets are quite large and continue to grow. Second, the curse of dimensionality is a critical problem in such datasets. For instance in a web log dataset, the dimensionality of the problem is equal to the number of web pages accessible in a given site. Third, the datasets are fraught with missing entries. Not all web pages are accessed by all customers, or in the e-commerce domain, not all products are purchased by all customers. Fourth, the predictions that are actionable have a very short lifetime. The need of the hour is for online predictive systems coupled with scalable offline mining techniques that can deal with large massively incomplete high dimensional datasets. We next discuss some of the related work in this area and discuss some of their pros and cons with respect to the above challenges.

2.1 Related Work

As mentioned earlier web navigation patterns show strong regularities which if discovered can give clues about users' interests and/or habits. This information is very valuable and can be used for cross-marketing, improving the performance of the web browser, personalization etc. To discover these patterns, several approaches such as collaborative filtering via clustering, Markov modeling of sequential navigation, and association mining have been suggested in the literature [13,17,8,21,12,14].

Collaborative filtering[20,7] uses consumer specific historical data to classify users into different groups such that each group has similar interests or buying patterns. Then it uses these clusters to predict a user's future interests by referencing his or her group. One limitation of this approach is that it can be quite inefficient especially for web sites where the number of physical web pages is quite large. Furthermore, this technique can lead to inaccurate predictions when user identities are hidden. The latter problem is true for all web sites that do not have a login-facility where users are inherently anonymous. While some recent approaches have been proposed to overcome these limitations based on the off-line clustering of page accesses (or page views) and the use of dimensionality reduction[1,19], these approaches are still quite expensive.

Compared with collaborative filtering which essentially clusters users, techniques that rely on sequential patterns contain more precise information about users' navigational behavior. Techniques that fall into this class include methods that rely on statistical models such as Markov models and sequential pattern mining [17,8,14,18,21]. These techniques have been well studied and it has been observed that while they are quite effective in making accurate predictions, the coverage of these approaches is not up to the mark[12]. Furthermore, these approaches also suffer from the problem that the number of rules generated by such models can be exceedingly high and therefore need (often expensive) pruning steps in order that they be effective[17,8,21,10].

Recently, the use of association rule patterns for web usage prediction has also been studied[12,13]. The coverage of these methods, unlike methods based on sequential patterns, is comparable to or better than collaborative filtering approaches[12]. Again like the sequential pattern approach, these approaches also suffer from the problem of generating an inordinately large number of rules which affects the scalability of online recommendation systems. Mobasher et al[12] propose a method to constrain the set of rules generated by the use of varying active session windows (user histories). A crucial difference between the approach we discuss in the next section and Mobasher et al's work[12] w.r.t to user histories is that they do not constrain the pattern generation phase whereas we constrain *both the pattern generation and the rule generation phases*. Furthermore, the rule constraining methods are quite different.

2.2 Frequent Pattern Mining

In this section, we briefly describe two of the central data-mining tasks related to our work, i.e., the discovery of association rules and the discovery of sequences. The discussion below follows [22] (associations) and [15] (sequences).

The problem of mining association rules over *basket* data was introduced in [2,3]. It can be formally stated as: Let $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$ be a set of m distinct attributes, also

called *items*. Each transaction T in the database $\mathcal{D}$ of transactions has a unique identifier, and *contains* a set of items, such that $T \subseteq \mathcal{I}$. An *association rule* is an expression $A \Rightarrow B$, where $A, B \subset \mathcal{I}$, are sets of items called *itemsets*, and $A \cap B = \emptyset$. Each itemset is said to have a *support* S if $S\%$ of the transactions in $\mathcal{D}$ contain the itemset. The association rule is said to have *confidence* C if $C\%$ of the transactions that contain A also contain B, i.e., $C = S(A \cup B)/S(A) * 100$, i.e., the conditional probability that transactions contain the itemset B, given that they contain itemset A.

Sequence Mining can be thought of as association mining over temporal datasets. A *sequence* is an ordered (over time) list of nonempty itemsets. A sequence of itemsets $\alpha_1, \ldots, \alpha_n$ is denoted by $(\alpha_1 \mapsto \cdots \mapsto \alpha_n)$. The *length* of a sequence is the sum of the sizes of each of its itemsets. The database is divided into a collection of customer sets where each customer set contains the set of transactions that customer is involved in order of occurrence. For a database D and a sequence α, the *support* or *frequency* of α in D is the number of customers in D whose sequences contain α as a subsequence. A rule $A \Rightarrow B$ involving sequence A and sequence B is said to have *confidence* c if $c\%$ of the customers that contain A also contain B.

Association patterns have the advantage that they can capture patterns relating to itemsets irrespective of the order in which they occur in a given transaction. For instance, one customer may access a certain web page A followed by another web page B; another customer may access B followed by A. Under association patterns the support (assuming that these are the only two transactions in the database) for the pattern AB is 100% whereas under sequence mining the support for $A \mapsto B$ and $B \mapsto A$ is only 50%. If the user sets the minimum support to 75% the pattern AB would not be found by the sequence mining approach.

A disadvantage of association mining is that it does not inherently use the notion of temporal distance which we believe is crucial for deciding which rules to apply for a given web transaction. Some variants of sequence mining do support it [6], but as far as we are aware no-one has evaluated this aspect within the context of web-log mining. We next describe an overview of our approach which combines elements of both the above approaches.

2.3 Overview

The essence of the solution proposed in this paper is to apply some novel variants of frequent pattern mining techniques on the problem at hand. The basic advantages of such techniques are that they are inherently scalable, can handle large growing datasets and are quite amenable to such high dimensional datasets with missing data.

We adopt a two phased approach in this work. In the first phase we are interested in determining the frequent usage patterns. These patterns will be utilized to develop the predictive model in the second phase. A key premise of our work is that *recently accessed web pages or recently purchased items have a greater predictive ability on future accessed web pages or purchases*. We embed this premise, in our approach, by proposing a method to temporally constrain the set of frequent itemsets. The resulting set of frequent itemsets differs from both traditional association patterns and sequential patterns. These distinctions will be detailed later. As a result of the first phase we are

able to effectively prune the set of usage patterns (when compared with a traditional association-based approach) without losing predictive accuracy (validated empirically).

The second phase involves building the predictive model. Traditionally frequent pattern algorithms use a support confidence framework to build a predictive model. As noted above, it has been observed by several researchers that the predictive model derived from association patterns tends to have greater coverage but also tends to be less precise for such domains. The problem with sequential patterns is that they tend to be narrow and more precise for such domains. Here we propose a hybrid strategy that combines both approaches while pruning the model size. This serves the dual purpose of achieving broad coverage and greater precision in predictions while also improving predictive efficiency. Our solution relies on the patterns generated from phase one. Instead of simply building a model based on the patterns we rely on the identification of active sequences for each pattern generated from phase one. The active sequences are used to prune or order the patterns according to their predictive ability. We show through detailed experimentation that the resulting model not only improves the accuracy but also reduces the model size improving prediction time significantly. In the next section we detail the proposed approach.

3 Constraining Association Patterns for Web-Log Mining

For the rest of this section we assume the following notation.

A web transaction, denoted by WT (analogous to the set of transactions T in the previous section), is composed of a sequence of web-page accesses (WA_i) where each web access is drawn from the set of all web pages (in a given site) WP (analogous to the set items I presented in the previous section. A web log (WL) can be represented as a set of web transactions. Formally these relations can be represented as:

$$WL = \{WT_1 \ldots WT_L\}$$

$$WT_i = \{WA_1 \ldots WA_T\}$$

$$WA_i \in WP$$

3.1 Incorporating Temporal Constraints

A problem with association mining approaches is that they often produce too many patterns. *The key is to identify a subset of these patterns that are really important and use them to build a predictive model.* In our case we would like to use these patterns to predict which web pages are likely to be accessed in the near future. Intuition tells us that more recently accessed web pages are likely to have a greater influence on what is likely to follow rather than web pages accessed at the start of a web transaction. One could capture this notion by constraining the set of patterns found such that not only do they occur frequently (exceed the user-specified minimum support) but also when they occur, they are constrained temporally.

A simple temporal constraint one might impose is the notion of a temporal window. Under this constraint we require that the items forming the itemset occur within a fixed-size temporal window within the transaction. We refer to such patterns as *windowed association patterns*.

A more relaxed temporal constraint might be to require that neighboring accesses of items within an itemset must not be too far apart. For example assume that there are three transactions in WL as shown below:

$$WT_1 = \{A, C, D, E, F\}$$
$$WT_2 = \{E, A, F, L\}$$
$$WT_3 = \{A, F, L, E, G\}$$

Under the classic definition of association patterns the pattern AFE has a support of 100% (ordering does not matter in association patterns). Under windowed association patterns, for a window size of three, AFE has a support of 33%. Under the neighbor constrained definition suppose the constraint imposed is that neighboring items should be within two accesses apart, then the support of said pattern is 66% (the first transaction violates it as the neighbors A and E are three accesses apart).

To formalize this we introduce the notion of *ranks* within a web transaction. The *rank* of a web page within a web transaction is the earliest position within that transaction corresponding to accessing that particular web page. Given two pages accessed (A, B) within a web transaction (WT) and associated ranks (say $rank(A \in WT) = i$ and $rank(B \in WT) = j$), the *rank-distance* between those two is simply the difference in the ranks (*rank-distance*$(A, B \in WT) = |i - j|$). More generally, given a transaction WT, an itemset K (involving k items) is found in that transaction if all of its items appear within the transaction. For each item in the itemset K, we identify its rank. Ordering the items according to their ranks, we now compute the rank-distance of consecutive neighbors. The maximum of these distances is taken as the rank-distance of that itemset given that particular transaction.

Using this definition we can rephrase the query in the example above to be: *Find all itemsets that exceed a pre-defined minimum support and that have a rank-distance of two or lower*. We refer to patterns generated by such constrained queries as *rank-distance association patterns*.

Some useful properties of rank-distance association patterns are:

- The set of frequent itemsets under rank-distance criteria is a subset of the set of frequent itemsets (with no rank-distance criteria).
- The set of frequent itemsets under rank-distance criteria does not uphold the anti-monotone (apriori) property that all of its subsets must be frequent and similarly constrained.
- The set of frequent itemsets under rank-distance criteria is a superset of the set of windowed frequent itemsets where window size is equal to the rank-distance.

The first and third properties are obvious and the second property is easy to prove. Assume a transactional database where each transaction contains exactly the same set of elements $\{A, B, C\}$ and in the same order. The itemset $\{ABC\}$ is frequent and satisfies the constraint *rank-distance*=1. The subset AC while frequent does not satisfy the above rank-distance constraint.

3.2 Identifying Active Sequences

In this section we discuss a method that can potentially combine the greater coverage provided by association rule mining methods with the greater accuracy provided by sequential pattern methods.

While rank-distance patterns are useful for capturing the intuition that recently accessed web-pages have a greater influence on prediction, it is also important to find out if a particular rank-distance association pattern is dominated by a single sequence (imposing a total order of the set of items composing the particular association pattern) or a set of sequences. We refer to such sequences as *active sequences*. For example consider an itemset (say $\{ABC\}$) that satisfies a certain rank-distance constraint and is dominated by a particular sequence that occurs 95% of the time (say $B \mapsto A \mapsto C$). Common sense dictates that using this pattern (ABC) to predict the next access of a transaction containing A followed by C is unlikely to be useful.

Activity ordering involves simply listing all possible sequences and ordering them by their activity (how often does a particular sequence occur). *Activity pruning* involves picking sequences in decreasing order of activity until the cumulative activity of sequences that have already been picked exceeds a user specified threshold (ρ). For example, let us say the sequence activity ordering for the association pattern ABC is as follows: $A \mapsto B \mapsto C : 90\%$, $B \mapsto A \mapsto C : 7\%$, $C \mapsto B \mapsto A : 3\%$, the remaining three sequences (0%). Let us assume the user specifies $\rho = 95\%$. The activity pruning procedure would then output the first two sequences as active sequences since the cumulative activity is 97%, which is greater than the ρ threshold.

The active remaining sequences after the *activity pruning* process can be used to either prune or re-order the association rules, which combines the greater accuracy of sequential pattern methods with the greater coverage of association mining methods. For the case of using active sequences for pruning, an association rule is pruned if it does not have a matched active sequence. For instance, in the above example, the rule $BC \Rightarrow A$ will be pruned because its corresponding sequences, i.e., $C \mapsto B \mapsto A : 3\%$ and $B \mapsto C \mapsto A : 0\%$, are eliminated by the *activity pruning* process when ρ is set to 95%. As for the other case that uses active sequences for re-ordering, association rules are ordered by their corresponding sequences' frequencies. If more than one sequence is available for a rule, the most frequent one is chosen. Both active sequence-based pruning and re-ordering are implemented in the predictive model, more details on them are explained in the next section.

3.3 Rule Generation

Once the set of rank-distance patterns and associated active sequences is determined, the next step is to derive the set of rules that will serve as a basis for the prediction model.

Under classical association rule mining, once the frequent patterns have been determined for each pattern of size k (involving k items), one can generate k rules involving a single consequent. Under rank-distance association mining, the set of frequent patterns is reduced (since it is a subset of the full set of frequent itemsets). This therefore reduces the set of rules that one will have to evaluate. Furthermore, for each pattern if there is one or more dominant active sequences one could replace the pattern by the set of rules

```
( 1 ) Algorithm RankDistanceApriori(itemsetTree, rdDist, winSize, minsupp)
( 2 )  //itemsetTree: contains all frequent itemsets mined by Apriori
( 3 )  //rdDist: maximum neighbor rank distance
( 4 )  //winSize: sliding window size
( 5 )  //minsupp: support threshold
( 6 ) begin
( 7 ) foreach itemset L in itemsetTree
( 8 )        set its support to 0; //initialize all rank-disntance associations
( 9 ) foreach transaction T in dataset{ //begin to compute rank-distance support
(10)    foreach itemset L in itemsetTree
(11)       if itemset L is contained in T {
(12)          if (neighborDist(T, L) ≤ rdDist) and (windowSize(L, T) ≤ winSize){
(13)                increase L's support by 1
(14)                get L's corresponding sequence in T;
(15)                increase this sequence's support by 1;
(16)          }
(17)       }
(18) } //End of computing rank-distance support
(19) foreach itemset L in itemsetTree
(20)       if (support(L) < minsupp) //support under rank-distance constraint
(21)          remove L from the itemsetTree; //remove itemsets that are not frequent
(22) generateActiveSequence(); //see text for detail
(23) generateRankDistanceRules(); //see text for detail
(24) end
```

Fig. 1. Rank-distance Association Mining Algorithm

dictated only by the active sequences. This further reduces the set of rules. Again, for a given frequent itemset if there are no dominant sequences we generate rules following the standard approach for association rules (as described in Section 2).

Figure 1 shows the pseudo-code of the algorithm that generates rank-distance association rules and active sequences.

Whichever method is used, once the rules have been generated they can be ordered according to their confidence and can be used for prediction as follows. When a new transaction is to be predicted, we first try to match the left hand side of each rule with the transaction (keeping in mind rank-distance criteria[1]). The list of matched rules are then used to predict the next access.

One has the choice to either prune rules based on active sequences (as described above) or order rules without pruning them (ordering the most active sequences first). The latter approach while not further reducing the set of rules (beyond the reduction obtained from rank-distance pruning) can yield better prediction accuracies.

[1] For instance if the transaction has involved accessing A, C, D, and E in that order, and the rank-distance criteria is one, then the rule $AE- > X$ does not match the given transaction even though A and E are present in the transaction.

Based on the above we have five possible strategies:

- **AR**: Prediction based on association rules (default)
- **Win-AR**: Prediction based on windowed association rules
- **RD-AR**: Prediction based on rank-distance association rules
- **RD-AR-ASP**: Prediction based on rank-distance association rules with active sequence pruning
- **RD-AR-ASR**: Prediction based on rank-distance association rules with active sequence re-ordering

Once the pruning has taken place, one is left with a final set of ordered rules which one can apply. We have a further user defined parameter which determines the size of our *prediction cache*. The prediction cache is defined as a set of possible predictions for a given transaction. Essentially if such a system is used on a proxy server then the proxy server can choose to prefetch as many web pages as the size of the prediction cache. The larger the prediction cache the greater the accuracy, in that at least one of the predictions will be correct. However, the larger the prediction cache the more the cost of prefetching. In our experimental results we will evaluate how choice of prediction cache size can affect accuracy.

4 Experimental Results

4.1 Dataset

The analysis presented in this paper is based on the MS Web log data obtained from the UCI repository[2]. The dataset consists of user log information (ordered accesses as recorded at the server end) over a one-week time-frame. For this web dataset the total number of transactions W_T is 32711. The total number of web pages accessed in the dataset W_P is 294. The average transactional length is 4.34 after excluding all the single-item transactions.

In order to build the prediction model and then evaluate it, we split the dataset into training and testing components. We generate association rules from the training data. Then we try to predict the last web page accessed in each transaction in the testing component.

In our experiments, we used 4 training and testing set splits as follows.

datasets	number of transactions
50:50 split	16457, 9545
60:40 split	19718 , 9070
70:30 split	22957 , 6801
80:20 split	26194 , 4594

Note that in each of the splits the number of transactions in the testing set is less than the expressed percentage of the training and testing split. This reflects the fact that

[2] http://kdd.ics.uci.edu/

a large number of the transactions involve a single item for which a partial transaction is not possible. Multiple instantiations of each training and testing split were produced at random and used in the experiments. All experiments report average case behavior for a given split over all instantiations.

4.2 Evaluation Criteria

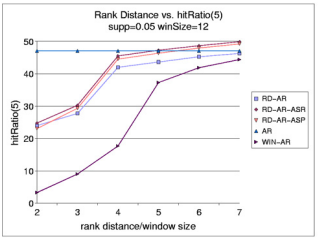
We define the following metrics to evaluate the performance of the various strategies.

- *Hit-Ratio*: Given a prediction cache size (PCS), a hit is declared if any one of the predictions in the cache is correct. The hit-ratio is the number of hits divided by the number of transactions in the testing component. This is our method of calculating the accuracy of a prediction model. We use the notation of hitRatio(PCS) to indicate the Hit-Ratio with a specific PCS.
- *Coverage*: Given a partial transaction in the testing data, it may be possible that none of the rules in the prediction model may apply. This is because there are no rules that may be applied to a given transaction. The percentage of transactions for which a prediction is possible is referred to as coverage. Coverage, can alternately be defined as the hit-ratio, given an infinite prediction cache.
- *Coverage-on-Predictable*: Defined as the ratio between hit-ratio and coverage. Basically, this measures, how close to an infinite prediction cache, a cache of fixed size performs.
- *Model-Size*: This parameter measures the number of rules generated and used by the predictive model. The larger the Model-Size the greater the online predictive time.

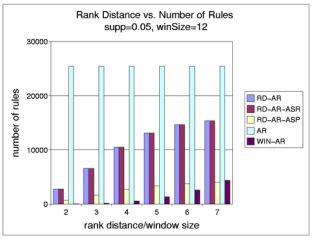
4.3 Analysis

The experiments involve two steps. The first step involves extracting the set of association patterns from the training data given a minimum support criterion, and rank-distance criteria where applicable. The second step involves taking the patterns generated from the first step and extracting rules according to the minimum confidence criteria and active sequence criteria where applicable. Once the rules have been generated and ordered they are used to predict the last access of each transaction in the testing data. A sequence-based predictive model is also built for the purpose of comparison. In this model, a sequence is defined as an ordered list of items with neighboring items occurring immediately next to each other in a transaction.

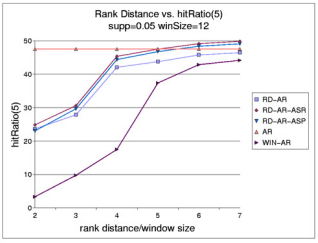
The sequence-based experiments are also carried out in two steps. The first step extracts all the sequences that meet the minimum support and minimum confidence from a training data set. The second step does the prediction, where the predictive model uses these extracted sequences to predict the last access of every transaction in the testing data set. The predictions for a testing transaction are ordered by the length of their corresponding sequences, where a prediction based on a longer sequence is preferred than a prediction based on a shorter one. Since the predictive model takes all extracted sequences when predicting the last access for a testing transaction, we refer this method as *All-sequence* in this paper.



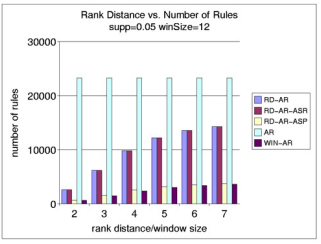
(a) HitRatio(5) on 60:40 split



(b) Model-Size on 60:40 split



(c) HitRatio(5) on 80:20 split



(d) Model-Size on 80:20 split

Fig. 2. Impact of Rank-Distance on Performance

Figures 2(a-d) show the impact of rank-distance (or window size for the windowed-AR method) on accuracy (hit-ratio) and model-size at 0.05% support. The main trend observed is that below a rank-distance of three, the association rule mining algorithm is performing much better accuracy-wise (see Figure 2(a) and Figure 2(c). At a rank-distance of four and beyond, the three RD-based approaches perform either comparably or do better. Moreover, this comparable performance is achieved with a much smaller Model-Size (8-20 times smaller in the case of RD-AR-ASP). This is extremely significant because the online-prediction time correlates strongly with model-size. The smaller the model-size the faster the online prediction time. While the model size of the Win-AR approach is comparable to RD-AR-ASP, the accuracy of this method does not approach the accuracy of the other four methods.

Amongst the rank-distance (RD) approaches it is clear that keeping track of active-sequences helps predictive accuracy. RD-AR-ASR always performs better accuracy-wise although the difference between RD-AR-ASR and RD-AR-ASP is usually quite small. The main reason is that the latter method prunes the rules from its model while the former simply re-orders the rules and hence has a more complete model that can predict future transactions (slightly) better. Also note that while RD-AR-ASR is more accurate, the model size for RD-AR-ASP is typically much smaller.

Figure 3 shows how different strategies affect the predictive time on the two dataset splits at two support levels:0.1% and 0.5%. In all cases, the RD-AR-ASP gives the best performance, followed by RD-AR, RD-AR-ASR and AR. Observe that although RD-AR-ASR can deliver the best predictive accuracy, it takes longer predictive time,

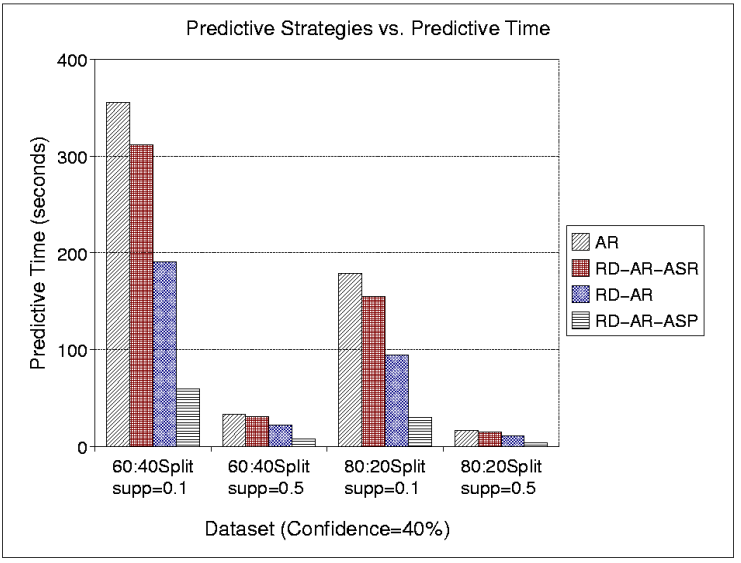
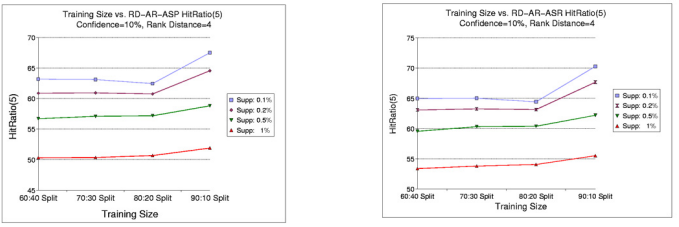


Fig. 3. Impact of Different Predictive Strategies on Predictive Time

this is because of the re-ordering step which improves the accuracy while at the same time slows down the predictive model's response. The overall trend also justifies the significance of using small predictive models for on-line prediction, as the smaller is a predictive model, the faster is the prediction.

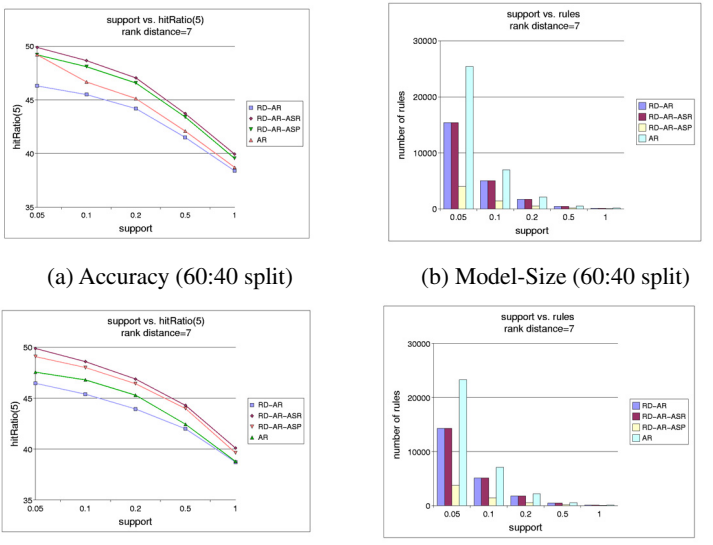


(a) RD-AR-ASP (b) RD-AR-ASR

Fig. 4. Impact of training set size on accuracy

Figures 4(a-b) depict the effect of training:testing splits on accuracy for the four support levels of 0.1%, 0.2%, 0.5% and 1%, where (a) shows the results from using the RD-AR-ASP predictive approach and (b) from the RD-AR-ASR approach. In both

cases, as expected, as the size of the training data is increased, the predictive model becomes more accurate. For this experiment we fixed the neighboring distance as 4 for the ranked-association approaches and the minimum confidence threshold at 10%.



(a) Accuracy (60:40 split) (b) Model-Size (60:40 split)

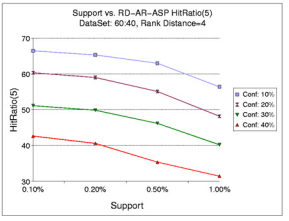
(c) Accuracy (80:20 split) (d) Model-Size (80:20 split)

Fig. 5. Impact of Support Parameter on Performance

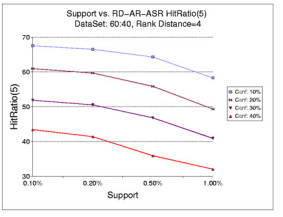
Figures 5(a-d) depict the effect of support on accuracy and model size for two dataset splits. At lower support values while the accuracy improves, the size of the model also grows. In fact, using the AR-based approach at 0.2% support one is able to get an accuracy of 43.5% with a model size of approximately 2000 rules. One can obtain much better accuracy (48.2%) at roughly the same model size (in fact slightly smaller) at 0.1% support using the RD-AR-ASP approach. The minimum confidence threshold is fixed at 40% for all these experiments.

Figures 6(a-d) evaluate the combined effect of support and confidence on predictive accuracy for two dataset splits, i.e., the 60:40 split and the 80:20 split. The main trend observed here is that at a given minimum support, as confidence decreases, the accuracy increases. We can see the accuracy is increased by 7-10% as the confidence is decreased by 10% in both RD-AR-ASP and RD-AR-ASR (See Figures 6(a-d).). The main reason of this increase is that a lower confidence provides a better coverage for the testing data as more rules are generated at a lower minimum confidence level. The rank distance is fixed to 4 in these experiments.

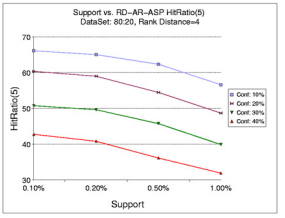
Figures 7(a-f) evaluate the effect of increasing the prediction cache size on accuracy and coverage. The first set of graphs (Figures 7(a-b)) compare the performance of the four configurations at low prediction cache sizes. On viewing the graphs for rank-distance



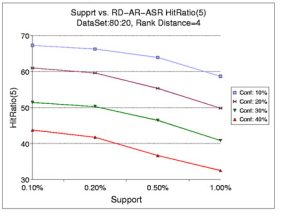
(a) RD-AR-ASP(60:40 split)



(b) RD-AR-ASR(60:40 split)



(c) RD-AR-ASP(80:20 split)



(d) RD-AR-ASR(80:20 split)

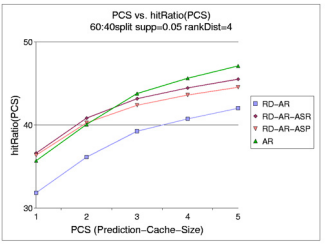
Fig. 6. Impact of Support and Confidence on Performance

of four, one observes an interesting trend. The AR method initially does worse than the RD-based methods that use active sequences, but as the the prediction cache size is increased, it does better. This is because the AR-based method has more rules it can apply to a given transaction and one of these other predictions often is useful resulting in a late hit. This clearly demonstrates that: i) RD-based methods capture the most useful rules (see how close RD-AR and AR follow one another); and ii) RD-based methods using active sequences always perform better when the number of predictions allowed is limited. On increasing the rank-distance parameter (Figure 7b) we see that the AR method always performs worse than the methods based on active sequences and that the RD-AR method again closely follows the AR method in terms of accuracy.

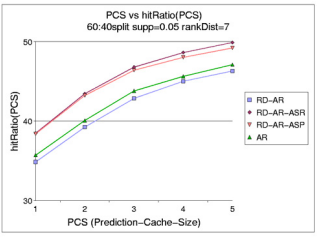
Figures 7(c-f) report the effect of prediction cache size on accuracy and coverage-on-predictable for two of database splits. Clearly at larger prediction cache sizes the accuracy improves as does the coverage-on-predictable. In fact, both approaches based on active sequences quickly converge asymptotically to a perfect coverage-on-predictable score (equal to 1).

Figures 8(a-f) and Figures 9(a-f) compare the predictive accuracy and model size between rank-distance association rules and sequences on two data splits with different minimum support/confidence levels and rank-distance criteria. Figures 8(a-f) show the results on the 60:40 split at support of 0.05% and 0.5%, and Figures 9(a-f) show the results on the 80:20 split at support of 0.1% and 1%.

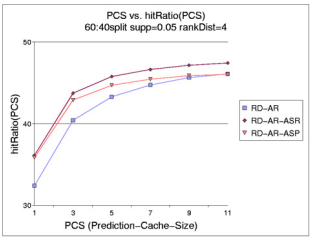
As shown in Figures 8(a) and (c) and Figures 9(a) and (b), the rank-distance association rule based methods, i.e., RD-AR-ASP and RD-AR-ASR, are more accurate than



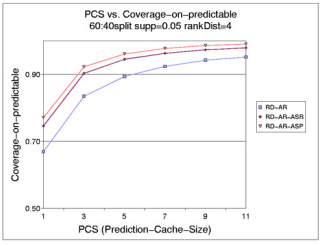
(a) Rank-Distance(4)



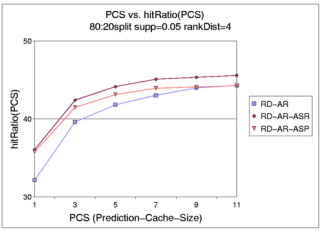
(b) Rank-Distance(7)



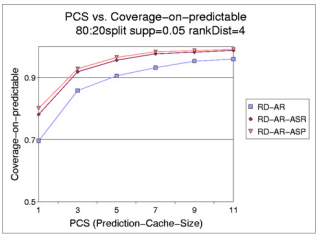
(c) Accuracy (60:40 split)



(d) Coverage-on-predictable (60:40 split)



(e) Accuracy (80:20 split)

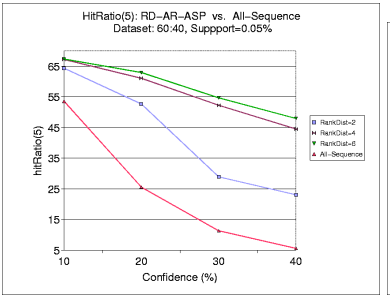


(d) Coverage-on-predictable (80:20 split)

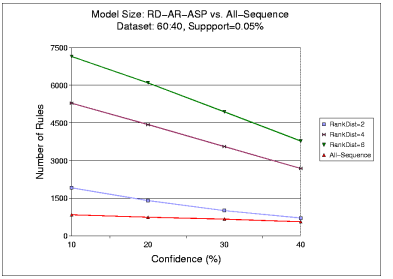
Fig. 7. Impact of Prediction-Cache-Size on Performance

the sequence-based at a lower confidence level even with a much constrained neighbor distance of 2 [3]. For the 60:40 split at the support of 0.05% and confidence of 10%, the hit ratio(5) of RD-AR-ARP, RD-AR-ASR and All-sequences are 64.38%, 65.90% and 55.70% respectively (See Figure 8(a).) However, as confidence goes up, the performance of the rank-distance rule based methods goes down. The main reason is that a higher minimum confidence threshold leads to a smaller set of rules, and accordingly a lower coverage. On the other hand, the all-sequence method shows much more vulnerability than the RD-AR based ones in terms of predictive accuracy as confidence level increases. As seen from the Figure 8(a), at confidence level of 40%, the all-sequence method gives an accuracy as low as one fourth of the lowest RD-AR based method. Two factors contribute to the all-sequence method's vulnerability. The first one is it has a

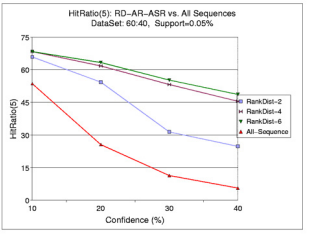
[3] By the definition of rank distance in Section 3, the neighboring distance of a sequence is 1.



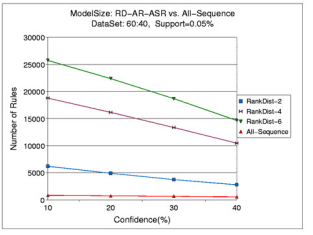
(a)HitRatio(5): RD-AR-ASP vs. All-sequence



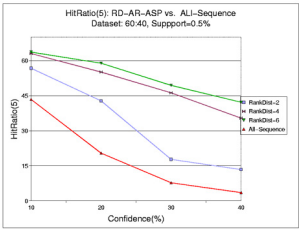
(b)Model Size: RD-AR-ASP vs. All-Sequence



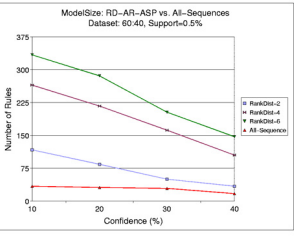
(c)HitRatio(5):RD-AR-ASR vs. All-sequence



(d)Model Size:RD-AR-ASR vs. All-Sequence



(e)HitRatio(5): RD-AR-ASP vs. All-sequence

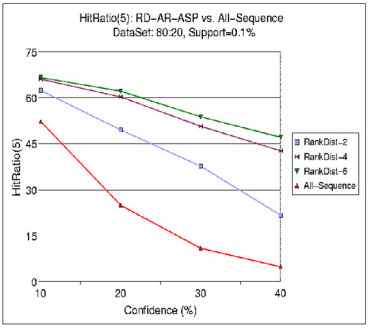


(f)Model Size: RD-AR-ASP vs. All-Sequence

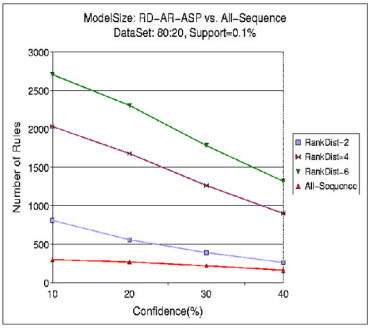
Fig. 8. Performance of rank-distance rules and sequences on 60:40 split

much lower coverage than the RD-AR based ones as the confidence level increases. The second one is because the restrictions imposed on sequences are much tighter than the RD-ARs, such as all items in a sequence must be totally ordered. This further shows the limitations presented in those sequence-based approaches and, to some extent, justifies why RD-ARs are more able to capture web usage patterns.

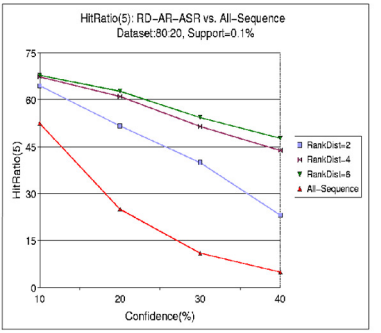
As for the model size, the RD-AR-ASR approach has a much larger size than the RD-AR-ARP and All-sequence, although it has the best predictive accuracy. The RD-AR-ASP method shows a much better performance than All-sequence when both the confidence and neighbor distance are set small, despite of a very similar model size. For



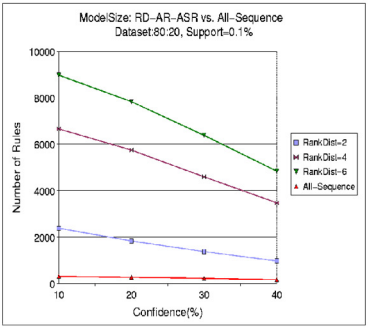
(a)HitRatio(5): RD-AR-ASP vs. All-sequence



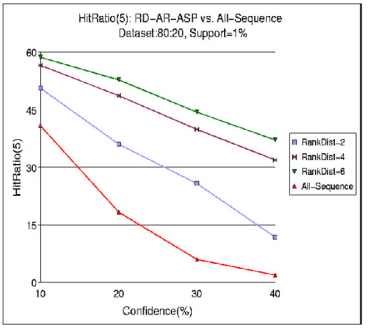
(b)Model Size: RD-AR-ASP vs. All-Sequence



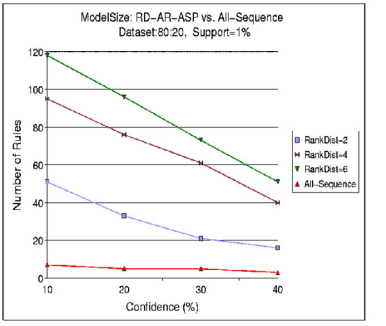
(c)HitRatio(5):RD-AR-ASR vs. All-sequence



(d)Model Size:RD-AR-ASR vs. All-Sequence



(e)HitRatio(5):RD-AR-ASP vs. All-sequence



(f)Model Size:RD-AR-ASP vs. All-Sequence

Fig. 9. Performance of rank-distance rules and sequences on 80:20 split

example, at confidence of 10% and neighbor distance of 2, for the 60:40 split, the hit ratio of RD-AR-ASP is 64.38% compared with 55.7% of All-sequence (see Figure 8(a)); and for the 80:20 split, the hit ratio of RD-AR-ASP is 62.45% compared with All-sequence's hit ratio of 53.35% (see Figure 9(a)). In both cases, the model size of RD-AR-ARP is close to the size of All-sequence. This feature is especially useful for online predictive applications, in the sense that it presents a better predictive accuracy with a smaller model size.

5 Conclusions and Ongoing Work

In this paper we presented a scalable approach for predicting web log accesses based on association rule mining. A problem with most association rule mining algorithms is that they often produce too many rules which can often lead to inaccurate predictions. In this article we present a set of strategies that temporally constrain and order the set of rules produced by such an algorithm. The intuition is that this temporal constraint actually captures the behavior of customers at a given web site. Results show that this approach not only reduces the rule set significantly but can often result in better predictions when the number of predictions permitted is limited. We are currently extending the analysis in several ways. First, we are performing similar analysis on other real web log datasets. Second, we plan to quantify the gains from generating a smaller model on online prediction strategies. Third, we are extending the idea in its basic form to the new domain of multi-dimensional spatial data analysis.

References

1. Charu C. Aggarwal, Joel L. Wolf, and Philip S. Yu. A new method for similarity indexing of market basket data. *Proceedings of the ACM SIGMOD Conference*, pages 407–418, 1999.
2. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Intl. Conf. Management of Data*, May 1993.
3. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In U. Fayyad and et al, editors, *Advances in Knowledge Discovery and Data Mining*. MIT Press, 1996.
4. Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.
5. Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. *Proceedings of the 20th International Conference on Very Large Data Bases*, 1994.
6. Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns: Generalization and performance improvements. *Proceedings of the Fifth International Conference on Extending Database Technology*, 1996.
7. John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998.
8. Mukund Deshpande and George Karypis. Selective markov models for predicting web-page accesses. *Proceedings SIAM International Conference on Data Mining (SDM'2001)*, April 2001.

9. Bernardo A. Huberman, Peter L. T. Pirolli, James Pitkow, and Rajan J. Lukose. Strong regularities in world wide web surfing. *Science*, 280(3):95–97, April 1998.
10. Ian Tian Li, Qiang Yang, and Ke Wang. Classification pruning for web-request prediction. *the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD'01*, 2001.
11. Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. *Knowledge Discovery and Data Mining*, pages 80–86, 1998.
12. Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Effective personalization based on association rule discovery from web usage data. *3rd ACM Workshop on Web Information and Data Management*, 2001.
13. Alexandros Nanopoulos, Dimitris Katsaros, and Yannis Manolopoulos. Effective prediction of web-user access: A data mining approach. *Proceedings of WEBKDD workshop*, 2001.
14. VN Padmanabham and JC Mogul. Using predictive prefetching to improve world wide web latency. *Computer Communication Review*, 1996.
15. S. Parthasarathy, M. Zaki, M. Ogihara, and S. Dwarkadas. Incremental and interactive sequence mining. ACM Confereince on Information and Knowledge Management (CIKM), Mar 1999.
16. Peter Pirolli and James Pitkow. Distribution of surfers' paths through the world wide web: Empirical characterization. *World Wide Web*, (2(1-2)):29–45, 1999.
17. James Pitkow and Peter Pirolli. Mining longest repeating subsequences to predict world wide web surfing. *Proceedings of (USITlsS'99: The 2nd USENIX Symposium on Internet Technologies and Systems*, 1999.
18. Ramesh R. Sarukkai. Link prediction and path analysis using markov chains. *the Ninth International World Wide Web Conference*, 1998.
19. Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender systems: a case study. *Proceedings of the WebKDD 2000 workshop at the ACM SIGKDD 2000*, 2000.
20. Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating 'word of mouth'. *Proceedings of the ACM CHI COnference*, 1995.
21. Qiang Yang, Haining Henry Zhang, and Tianyi Li. Mining web logs for prediction models in www caching and prefetching. *the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD'01*, 2001.
22. M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New parallel algorithms for fast discovery of association rules. *Data Mining and Knowledge Discovery: An International Journal*, to appear, December 1997.

Mining WWW Access Sequence by Matrix Clustering

Shigeru Oyanagi[1], Kazuto Kubota[2], and Akihiko Nakase[2]

[1] Ritsumeikan University
oyanagi@cs.ritsumei.ac.jp
[2] TOSHIBA Corp.
{kazuto,nakase}@isl.rdc.toshiba.co.jp

Abstract. Sequence pattern mining is one of the most important methods for mining WWW access log. The Apriori algorithm is well known as a typical algorithm for sequence pattern mining. However, it suffers from inherent difficulties in finding long sequential patterns and in extracting interesting patterns among a huge amount of results.
This article proposes a new method for finding generalized sequence pattern by matrix clustering. This method decomposes a sequence into a set of sequence elements, each of which corresponds to an ordered pair of items. Then matrix clustering is applied to extract a cluster of similar sequences. The resulting sequence elements are composed into a generalized sequence.
Our method is evaluated with practical WWW access log, which shows that it is practically useful in finding long sequences and in presenting the generalized sequence in a graph.

1 Introduction

In view of the rapid increase of e-commerce on the World Wide Web, marketing strategy is shifting toward mass customization. The application of data mining methods on the WWW access log (Web Usage Mining) is expected to provide a general and powerful method for customization of e-commerce. Now, Web Usage Mining is recognized as a killer application of data mining technologies[18].

A general method for Web Usage Mining at first generates a session data from access log file by grouping log records of the same user within a reasonable time interval, and then applies various mining algorithms on the session data.

Mining sequential pattern is well known as one of the most important methods for Web Usage Mining. Basic algorithm for mining sequential pattern is introduced by R.Agrawal and R.Srikant [5][6], which is the application of the Apriori algorithm for association mining[4].

The Apriori algorithm adopts multiple-pass candidate generation and test approach in sequential pattern mining. This is outlined as follows. The first scan finds all of the frequent items which form the set of single item frequent sequences. Each subsequent pass starts with a seed set of sequential patterns, which is the set of sequential patterns found in the previous pass. This seed

O.R. Zaïane et al. (Eds.): WEBKDD 2002, LNAI 2703, pp. 119–136, 2003.

set is used to generate new potential patterns, called candidate sequences. Each candidate sequence contains one more item than a seed sequential pattern, where each element in the pattern may contain one or multiple items. The number of items in a sequence is called the length of the sequence. So, all the candidate sequences in a pass will have the same length. The scan of the database in one pass finds the support for each candidate sequence. All of the candidates whose support in the database is no less than minimum support form the set of the newly found sequential patterns. This set then becomes the seed for the next pass. The algorithm terminates when no new sequential pattern is found in a pass, or no candidate sequence can be generated.

The Apriori algorithm generates all the sub-sequences as intermediate results to find a long sequence. For example, suppose there is only a single sequence of length 100 in the database, and the minimum support is 1. The Apriori algorithm must generate 100 length-1 sub-sequences, 100 * 100 length-2 sub-sequences, and so on. The total number of generated sub-sequences is greater than 10^{30} [8]. Since the total execution time is proportional to the number of generated sub-sequences, it is practically impossible to find a long sequence with small minimum support.

There are a lot of studies about efficient mining of sequential patterns or other frequent patterns at time-related data[3]. Almost all of the previously proposed methods for mining sequential patterns are based on the Apriori algorithm. Therefore, these methods have the common problems such as efficiency and difficulty in finding long sequential patterns. And also there are some studies about the improvement of the Apriori algorithm, mainly with respect to the efficiency in finding long sequential patterns[8][9] [10]. However, these approaches use step-wise generation of candidate sequences as same as the Apriori algorithm. Although the execution time can be reduced from original Apriori algorithm, they still require longer execution time for generating longer sequences.

Our motivation is to develop sequence mining algorithm for the following applications.

1. To provide information for WWW site managers about page usage. Namely, presenting a typical page access sequence which includes a specified page.
2. To personalize the WWW pages for users. Namely, recommending favorite pages for a user by analyzing his access sequence.

These applications require that sequence mining should be efficient enough for real time execution, the result of mining should be easy to understand, and the execution time should not depend on the length of the sequence.

This article proposes a new method for mining sequential patterns whose problem definition is different from the Apriori algorithm. Our approach aims to find a super-sequence by generalizing a set of similar sequences. We must notice that a super-sequence is not an actual sequence appeared frequently in sequence database. Instead, a super-sequence is a virtual sequence which can represent characteristics of a set of similar sequences. The obtained super-sequence can present a global structure of a set of similar sequences, which makes it easier to analyze the mining result.

Our method is based on matrix clustering[19][20], which is our original method for finding relationships between two independent entities. Matrix clustering generates dense sub-matrix from a large-scale sparse matrix by exchanging the order of rows and columns. We have shown that matrix clustering is powerful in finding clusters of users and pages from WWW access log. In addition, we have shown that matrix clustering can be applied to WWW access prediction[20]. The basic representation of matrix clustering is a binary matrix of two entities, which cannot represent sequences directly. In this article, we introduce a new method to represent a sequence into matrix. It decomposes a sequence into a set of sequence elements, each of which corresponds to an ordered pair of items. Then, we can represent the relations of each sequence and correspondent sequence elements involved in a sequence into a binary matrix, and then matrix clustering can be applied to extract a cluster of sequences and sequence elements. The resulting sequence elements involved in the cluster are composed into a super-sequence.

Advantages of this method are described as follows.

1. Inserting lacked items
   Suppose four sequences (A,B,C), (A,B,D), (A,C,D) and (B,C,D) as an example. Each of these sequences lacks one item from the base sequence (A,B,C,D). Conventional sequence mining algorithm cannot produce (A,B,C,D) from these four sequences, but produces sub-sequences (A,B), (A,C), (A,D), (B,C), (B,D), and (C,D) which are common in these sequences as a result. On the other hand, our method can generate (A,B,C,D) as a super-sequence obtained by composing sequences (A,B,C), (A,B,D), (A,C,D) and (B,C,D).
   We think that generalization of sequences corresponds to find a hidden structure among a set of similar sequences. Inserting lacked items is a fundamental function to find a generalized sequence. Our method can insert lacked items by decomposing a set of similar sequences into sequence elements, and composing them after matrix clustering. Obviously, the length of the super-sequence tends to be longer than that of the sequences found by the conventional algorithm.
2. Efficient execution
   Since the Apriori algorithm finds all the sequences whose frequencies are no less than minimum support, its execution time becomes huge when the minimum support is small. Moreover, the execution time grows as the length of the sequence becomes longer. This comes from the multiple-pass generation of candidate sequences of the Apriori algorithm. On the other hand, matrix clustering directly generates a super-sequence without iterating stepwise generation. So, the execution time doesn't depend on the length of the super-sequence.
   In addition, the algorithm of matrix clustering, which is named Ping-pong algorithm as described in Section 2, is efficient enough for large sparse matrix, which enables real-time processing for personalization.

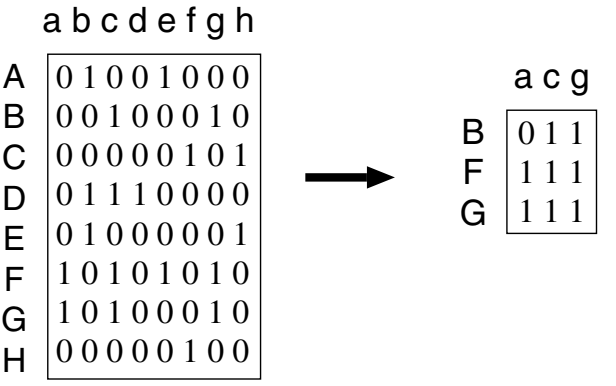


Fig. 1. Matrix Clustering

3. Visualizing the sequence structure
   It frequently happens that a large set of similar sequences, each of which differs slightly are obtained by conventional sequence mining. Since the mining result is given as a long list of sequences, it is difficult to understand the global structure hidden in the set of similar sequences.
   Obviously, a super-sequence can present the global feature more clearly than the list of sequences. In addition, a super-sequence can be represented into a graph structure. This graph makes it easy to understand the global structure visually.

This article consists of the following sections. Section two introduces matrix clustering and the Ping-pong algorithm. Section three describes the problem specification of the proposed method for mining sequential patterns. Section four discusses the characteristics of the proposed mining method in detail. Section five discusses evaluation of the proposed method by experiment with actual WWW access log. The result is also compared with that by conventional methods.

2 Matrix Clustering

A target matrix for matrix clustering is shown in Fig.1. In Fig.1, we assume that the order of rows and columns is meaningless, so we can exchange the rows and columns arbitrarily. Matrix clustering is a method to extract a dense sub-matrix from the base matrix by exchanging rows and columns. For example, Fig.1 shows that a dense sub-matrix which consists of rows B,F,G and columns a,c,g is extracted. A user specifies seeds for clustering (rows or columns) and constraints (size and density) of the resulted sub-matrix as parameters.

We have proposed a fast algorithm named the Ping-pong algorithm for matrix clustering which is intended to reduce the execution time by utilizing the sparseness of a base matrix. The Ping-pong algorithm uses marker propagation.

Marker propagation is generally defined as a computation model to represent a unit of processing as a node and a relation between nodes as a link. It activates processing by passing a marker from an activated node through a link successively. In the Ping-pong algorithm, rows and columns are represented as nodes. When a matrix element is 1, the corresponding row and column are connected by a bi-directional link. When a node receives a marker, it is accumulated, and the total count of the received markers is used for pruning. The algorithm iterates marker propagation between rows and columns until the state where the activated rows and columns are not changed. The processing of the Ping-pong algorithm is explained with Fig.2.

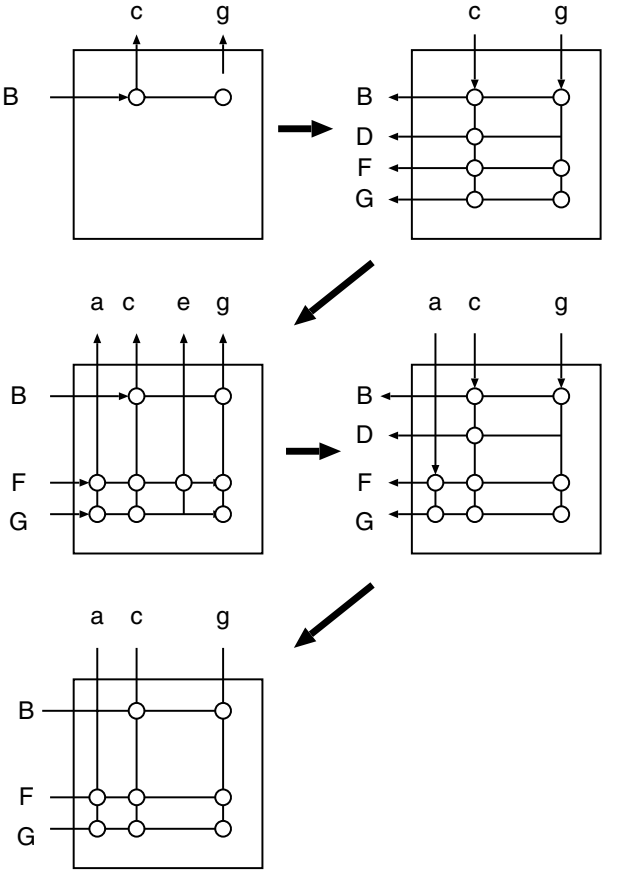


Fig. 2. Ping-pong algorithm

The Ping-pong algorithm starts with the specification of the starting row as row_B. At first, row_B is activated and it sends a marker to the linked

columns(c,g). Next, pruning of columns is performed. Since the counts of received markers are 1 for columns(c,g), all of them are activated without pruning. Then, activated columns(c,g) propagate markers to linked rows. For example, column_c sends a marker to rows(B,D,F,G) and column_g sends a marker to rows(B,F,G). Then, each row calculates the count of received markers. For example, row_B, row_F and row_G have received 2 markers at this step, nut row_D has received 1 marker. Then the pruning of rows is performed. In this case, row_D is pruned and the other rows(B,F,G) are activated. Next, rows(B,F,G) send markers and columns(a,c,e,g) receive markers. Since column_e received only 1 marker, column_e is pruned and the other columns(a,c,g) are activated. Finally, rows(B,F,G) and columns(a,c,g) are activated to form the result.

In the Ping-pong algorithm, area and density of the sub-matrix can be easily calculated by the activated rows/columns and number of markers. Therefore, the threshold value at the pruning can be dynamically calculated so as to satisfy the constraint of area and density.

The number of iterations in the Ping-pong algorithm is constant(i.e less than 10 even in a large scale matrix). Hence, the computational complexity of the Ping-pong algorithm is only dependent to the number of marker propagations, which corresponds to the size of resulted sub-matrix.

There are similar methods to matrix clustering which use matrix representation, such as collaborative filtering[14][15] and co-clustering[16][17]. Collaborative filtering represents the relationship between customers and items in multiple value matrix, to recommend items for current user. The algorithm is to find similar customers first, and then generate items based on the difference between current user and similar users. Co-clustering represents the relationship between documents and keywords in binary matrix, to cluster documents and keywords by dividing the total matrix. The major difference between these methods and matrix clustering lays in its algorithm. Although the problem representation of matrix clustering is similar to these methods, our original ping-pong algorithm enables to find a dense sub-matrix from a given seed on real time.

3 Problem Definition

In this section, we define the problem of our sequence mining method. First, we explain the problem definition for the Apriori algorithm, and then explain the modification for our method.

The problem definition for the Apriori algorithm is as follows. Let I=i1,i2,...,in be a set of all items. An itemset is a subset of items. A sequence is an ordered list of itemsets. A sequence is denoted by (s1,s2,...sj), where sj is an itemset. An item can occur at most once in an element of a sequence, but can occur multiple times in different elements of a sequence. The number of instances of items in a sequence is called the length of the sequence.

Next, we explain the modification of the above problem definition for our method. Since we restrict the application to WWW access sequences, we simplify the problem definition as follows.

1. A sequence is defined as an ordered list of items. Namely, an element of a sequence is not an itemset but an item. The problem definition of the Apriori algorithm requires handling of simultaneous occurrence of multiple items which are necessary for application to basket analysis in the retail application. However, in WWW access log, all the WWW page accesses are ordered serially, hence we can simplify the problem definition.
2. An item can occur at most once in a sequence, namely an item cannot occur multiple times in a sequence. This restriction is caused by representing the relationship between sequences and sequence elements in a binary matrix. Practically, there are multiple occurrences of items in a sequence frequently. We only consider the first occurrence of the same item in a sequence, and the following occurrences are removed. This restriction is introduced in order to focus the analysis of access sequences on WWW personalization.
   For example, a sequence (a,b,a,b) involves multiple occurrences of items a and b. Then, the correspondent sequence elements are (a,b), (a,a), (b,a) and (b,b). We focus the order of first appearance for each pair of items, namely we want to know which page influences other pages. Hence, the sequence (a,b,a,b) is reduced to (a,b) by removing the second occurrences for each item, and the corresponding sequence element is (a,b), which means that item a influences item b.

Next, we will discuss the definition of a super-sequence from the structural meaning and statistical meaning.

Structure of a Super-Sequence. A super-sequence is a structure composed of a set of sequences. The simplest structure of a super-sequence is a sequence obtained by concatenating multiple sequences, such as (A,B,C,D) from concatenating (A,B,C) and (B,C,D). The structure of a super-sequence is not limited to a simple sequence, but involves a branch and a join. For example, two sequences (A,B,C,E) and (A,B,D,E) can be generalized into a super-sequence (A,B,[C,D],E) which has a branch at B and a join at E. This implies that a super-sequence should be defined as a directed graph whose node corresponds to an item and whose link corresponds to an order between nodes.

In this article, we restrict that a super-sequence does cot include a loop. For example, sequences of (A,B,C), (B,C,A) and (C,A,B) involves a loop. In this case, we remove loops by cutting the links B-A, C-A and C-B off after performing matrix clustering.

Statistical Meaning of a Super-Sequence. It is necessary to define a super-sequence not only by its structure but also by its statistical property. Here, we define density of a super-sequence as follows.

$$density = \frac{\sum_i frequency\_of\_link(i)}{total\_kinds\_of\_links \times total\_sequences}$$

For example, when the set of sequences are completely the same, the generated super-sequence is also the same as the sequences, and the density of the generated super-sequence is 1. It means that the density corresponds to the similarity between a set of sequences. We also notice that the density is equal to the density of a matrix whose row corresponds to a sequence and column to a ordered pair of items in a sequence.

Problem Definition. The problem to find a super-sequence from a given set of sequences is described as follows.

1. Find a super-sequence which is generated by a set of similar sequences with a given sequence and whose density is greater than a given value.
2. Find a super-sequence which is generated by a set of similar sequences with a given ordered pair of items and whose density is greater than a given value.

Problem 1 is for personalization in which a given sequence corresponds to a current user's behavior. Problem 2 is for site manager in which a given ordered pair of items corresponds to a focused pages that he want to know.

Both of these problems can be executed with the same algorithm but only different at the initial value for matrix clustering.

4 Method

4.1 Basic Algorithm

The basic algorithm for sequence mining with matrix clustering consists of the following three steps: (step 1) a step to generate binary matrix from the session database, (step 2) a step to obtain a dense sub-matrix by applying matrix clustering, and (step 3) a step to generate a super-sequence from the obtained sub-matrix.

In step 1, the sequence database is transformed into a binary matrix whose row corresponds to each sequence and whose column corresponds to an ordered pair of pages. An ordered pair of pages (x,y) represents that page-x appears earlier than page-y in the same sequence. Since the total number of ordered pairs of pages should be huge in a large WWW site, it is necessary to restrict handling of ordered pairs of pages whose occurrences are not less than minimum support.

In step 2, matrix clustering is applied to the base binary matrix. The resulting dense sub-matrix represents the relationship between a group of sequences and a group of ordered pairs of pages which are related each other.

In step 3, the group of ordered pairs of pages is synthesized into a super-sequence which is represented in a graph. The algorithm is as follows. First, count the number of source and destination for each page by counting the position appeared in the ordered pairs of pages. Then, select pages whose destination count is 0. If it is not found, then a loop must be included. So, select a node with

minimum destination count in order to remove a loop. The selected pages are put at the leftmost position in the graph. Then, decrement destination count of each page by the appearance of the ordered pairs of pages whose sources are the selected pages, and continue this procedure until all the pages are put into the graph. Finally, links are put for each ordered pairs of pages. This graph representation is much easier to understand than representation of a list of all frequent sequences. Hence, a user can easily analyze the result of sequence mining.

4.2 Example

An example is given in Fig.3. Fig.3(1) shows that 10 sequences (s1-s10) which access five pages (A,B,C,D,E).

In this case, the total number of ordered pairs of pages is 20, so all the ordered pairs are selected into the base matrix without pruning. Therefore, the size of base matrix is 10 x 20, For example, sequence S1 represents the page access sequence of A-C-D. The ordered pairs of pages included in S1 are (A,C), (A,D) and (C,D). and the corresponding elements in the matrix are set to 1 as shown in Fig.3(2).

In the next step, matrix clustering is applied with specifying S1 as an initial value. Then, a dense sub-matrix with rows(S1, S2, S3, S4, S5) and columns(AB, AC, AD, BD, CD) is obtained as shown in Fig.3(3). Different sub-matrix can be obtained by specifying S6 as an initial value. In this case, the sub-matrix consists of rows (S6, S7, S8, S9, S10) and columns (BA, BC, BE, CA, CE) as shown in Fig.3(4).

These columns in the sub-matrix are combined into a super-sequence represented in a graph. The graph generation algorithm is shown in Fig.4. The first sub-matrix consists of five sequence elements (A,B), (A,C), (A,D), (B,D) and (C,D). A node table is generated by counting the appearance of each item at the position of the sequence elements in the resulted sub-matrix. Row-A in the node table has value 3 at the source column and 0 at the destination column. It means that node-A appears three times at the first position of the column in the first sub-matrix such as (A,B), (A,C), (A,D), and zero times at the second position. Also, Row-B means that node-B appears once at the first position as (B,D), and once at the second position as (A,B). After creating the node table, rows which has zero value at the destination column are selected, and their ranks are set to 1. Then, the node table is updated by removing the selected rows and by decreasing the count of source and destination columns which correspond to the selected nodes. In this example, node-A is selected and the node table is updated as shown in Fig.4(2). This step is continued until all the nodes are selected, and finally each nodes is given its rank. Then, nodes are allocated from left to right ordered by the rank, and links are drawn between nodes which correspond to the sequence elements. Finally, the generated graph is shown in Fig.4(4), and the second sub-matrix is converted into a graph as shown in Fig.4(8). In this figure, we neglect the links for simplicity which can be obtained by applying transition rule on the graph(i.e. link A-D is not shown because it can be obtained by composing link A-B and link B-D).

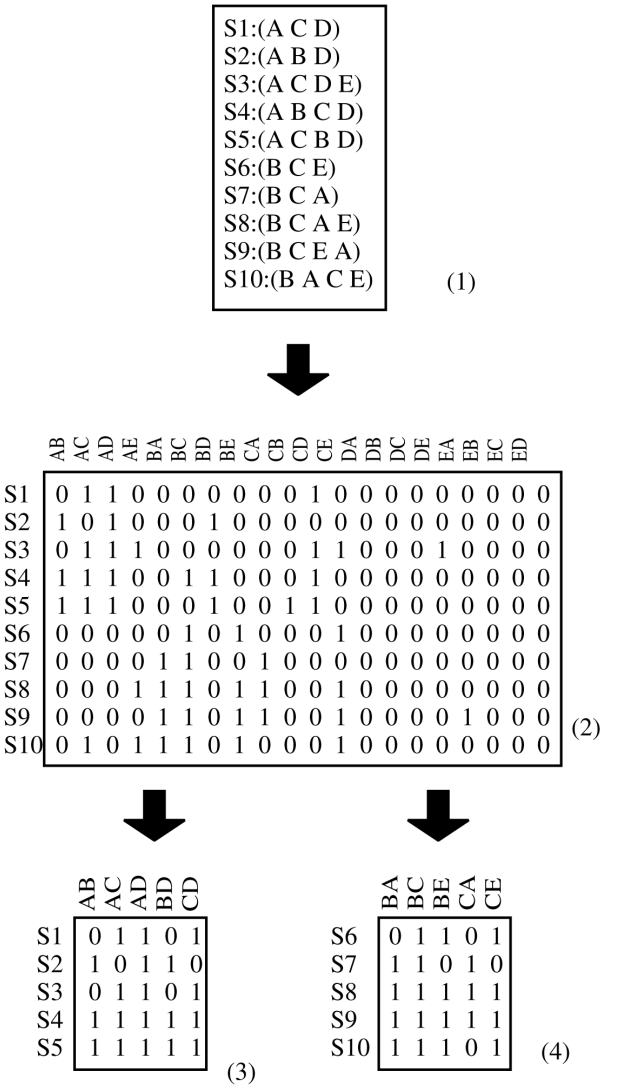


Fig. 3. Example of sequence finding by matrix clustering

Now, we compare the result between our method and the Apriori algorithm. The result of the execution of the Apriori algorithm with the same data set is shown in Fig.5. Here, we assume that the minimum-support is 3. Fig.5 shows that the Apriori algorithm can generate four sequences (A,B,D), (A,C,D), (B,C,A) and (B,C,E) with length 3. They are just the same as the result of the proposed method which can be derived from the graph as shown in Fig.4. Combining

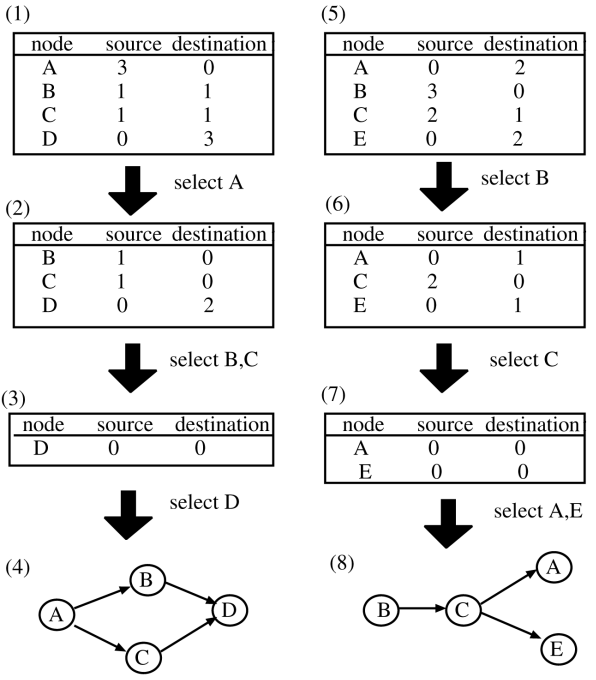


Fig. 4. Example of graph generation

these four sequences into a graph results in a complex structure including loops as shown in Fig.6, which makes it difficult to understand the common features included in these sequences.

It should be noted that these four sequences can be divided into two groups, each of which corresponds to the sub-matrix obtained by specifying appropriate initial value at matrix clustering. We can combine sequences in each group separately, and generate graphs easily by the proposed method. On the other hand in the case of the Apriori algorithm, these four sequences cannot easily be divided into groups, because of the lack of information to divide them.

4.3 Characteristics of a Super-Sequence

We discuss the characteristics of the super-sequence. Here, we assume that the frequency of each session is 1.

(1) Capability of Inserting Lacked Items. A density is specified to generate a super-sequence. The density means the similarity among a set of sequences which form a super-sequence. When the density is 1, the set of sequences are completely the same, and the composed super-sequence is also the same as each sequence.

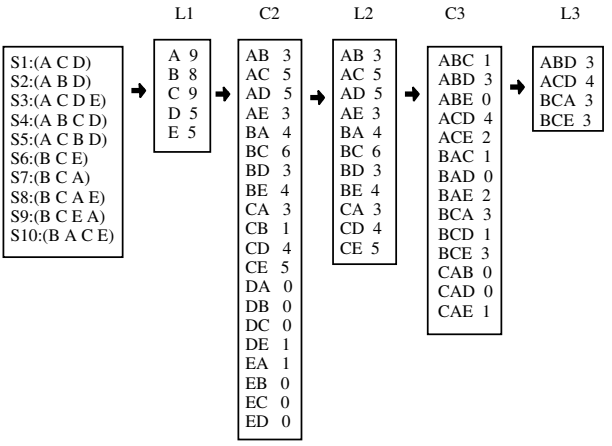


Fig. 5. Example of sequence finding by the Apriori algorithm

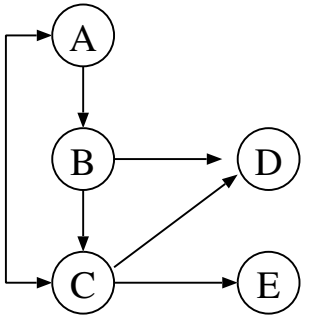


Fig. 6. Generating a graph from sequences by the Apriori algorithm

Here we assume that the super-sequence is a simple sequence, which involves no branch and join. If there is a lack of items in each sequence, proposed method has a capability to insert the lacked items to generalize sequences. The capability of inserting lacked items depends on the specified density. Namely, the density corresponds to the ratio of lacked items. If we specify low density at executing our method, then the resulted super-sequence may involve a lot of lacked items.

We give a simple example in Fig.7. Suppose that there are five WWW pages(A,B,C,D,E) and five sessions S1(A,B,C,D), S2(A,B,C,E), S3(A,B,D,E) S4(A,C,D,E) and S5(B,C,D,E), each of which lacks one page access. Matrix clustering can generate a dense sub-matrix by specifying one of the pages as an initial value as shown in Fig.7. Note that the density of the matrix is 0.6 (which means Permutation(4,2) / Permutation(5,2)). The columns in the dense sub-matrix can be synthesized into the sequence (A,B,C,D,E) of length five. It is

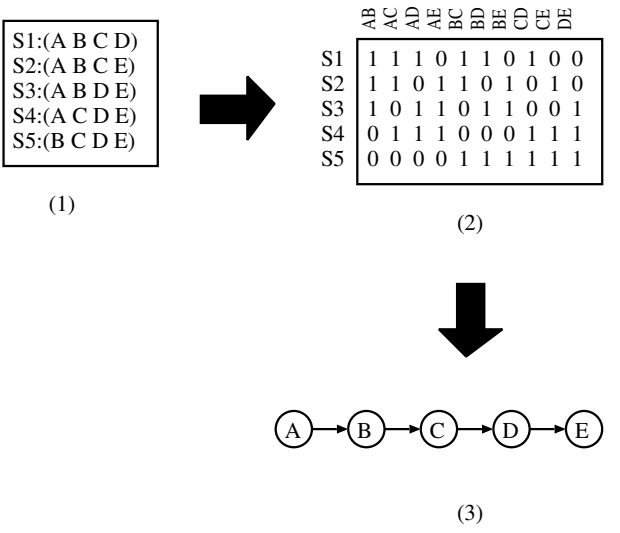


Fig. 7. Finding a long sequence by the proposed method

apparent that this is a super-sequence of the given five sequences, and the proposed method can reorganize the super-sequence from lacked sequences easily. In general,
$density = Permutation(n - x, 2) \div Permutation(n, 2)$
where n is the length of super-sequence and x is the number of lacked items in a sequence. It means that our method can insert more than 20 % of lacked items when the super-sequence is long by setting density to 0.5.

Fig.8 shows the execution of the same data by the Apriori algorithm with minimum-support 2. We can obtain 10 sequences of length 3. Namely, it is possible to find common sub-sequences by the Apriori algorithm, but it is difficult to find a super-sequence by the Apriori algorithm. This characteristics of the proposed method is quite suitable to find a long sequence.

(2) Connectivity of a Super-Sequence. A super-sequence is connective when there exists a path between every pair of nodes on the graph. If a super-sequence is divided into two sub-graphs with no path across them, then the super-sequence is not connective. We give an example of non-connective super-sequence and its matrix representation in Fig.9. Fig.9 shows that the density of the non-connective super-sequence is less than 0.5 when it consists of two sequence with the same length. If the lengths of the two sequences are different, then matrix clustering tends to prune rows and columns that correspond to the shorter sequence, and the resulted sub-matrix represents the longer sequence only. Hence, our method guarantees the connectivity of the super-sequence by specifying the density at greater than 0.5.

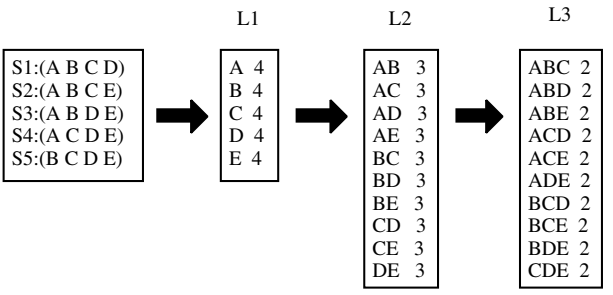


Fig. 8. Finding a long sequence by the Apriori algorithm

Fig. 9. non-connective super-sequence

(3) Structure of a Super-Sequence. Now we consider the relationship between density and branch and join. Fig.10 shows some examples of super-sequences including branch. In Fig.3(a), the top node has a branch, and the lengths of the path from the branch are the same. Then the density of the super-sequence is 0.5 as same as the non-connective case. We can observe that the corresponding matrix is divided into two sub-matrices of equal size.

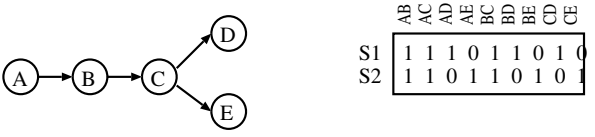
In Fig.3(b), branch occurs at the last node. Then the density of the super-sequence is 2/3. The difference of the density comes from the ratio of shared links among sequences. So, when the sequences are combined into a tree structure, no links are shared among sequences from a root to a leaf. Hence, our method does not generate the total tree structure. The resulted super-sequence depends on the seed for matrix clustering. If we specify a leaf of the tree as a seed, then a path from the root to the leaf is generated. And when we specify the root of the tree as a seed, then the most frequent path to a leaf is generated.

These characteristics say that our method is suitable for generating a long sequence which can be obtained by inserting lacked items. We think that the appropriate density for proposed method is greater than 0.5.

5 Experiment with Practical Data

The proposed method was evaluated with practical WWW access log. The experiment is performed in order to compare the proposed method with the Apriori algorithm. Data used in the experiment are given in Table 1.

(a) branch at the top of the sequence



(b) branch at the last of the sequence

Fig. 10. super-sequence including branch

Table 1. Data size for experiment

total records	180,249
total sessions	39,453
total pages	4,866
average session length	4.6

In the Apriori algorithm, minimum-support was set to 10. Table 2 shows the result of the Apriori algorithm. The maximum length of the resulting sequence was 7, and 22 kinds of sequences with length 7 were obtained by the Apriori algorithm as shown in Table 3.

Table 2. Number of sequences generated by the Apriori algorithm

sequence length	frequency
1	82
2	1158
3	2547
4	2111
5	899
6	211
7	22

Table 3 shows that most of the sequences with length 7 are similar each other, and it is expected that they can be further generalized into a super-sequence.

Next, we explain the experiment of the proposed method. We generated binary matrix with row for each session and column for each ordered pair of pages whose minimum support is 10. The execution of matrix clustering is iterated for each column as an initial value. Parameters used in the experiment of matrix clustering are given in Table 4.

Table 3. Sequences of length 7 generated by the Apriori algorithm

sequences	frequency
492 742 810 832 835 862 878	15
492 742 832 835 862 878 876	11
492 742 834 835 862 878 876	11
493 810 832 835 862 878 876	10
742 748 810 832 835 862 878	19
742 748 810 832 835 878 876	10
742 748 832 835 862 878 876	11
742 748 834 835 862 878 876	11
742 810 831 835 862 878 876	10
742 810 832 835 848 862 878	12
742 810 832 835 848 878 876	10
742 810 832 835 862 876 869	12
742 810 832 835 862 878 493	20
742 810 832 835 862 878 748	16
742 810 832 835 862 878 852	10
742 810 832 835 862 878 859	19
742 810 832 835 862 878 865	12
742 810 832 835 862 878 868	14
742 810 832 835 862 878 869	13
742 810 832 835 862 878 876	33
742 810 832 835 862 878 894	13
748 810 832 835 862 878 876	11

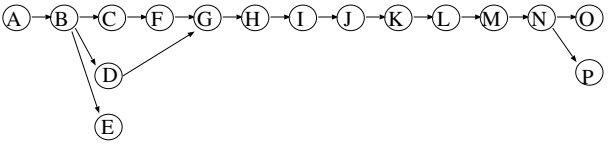
The maximum length of the resulting super-sequence was 16, much longer than that obtained by the Apriori algorithm. Fig.11 shows the generated graph from the maximum length super-sequence. We can see that all of the 22 sequences of length 7 which are obtained by the Apriori algorithm are included in this graph.

Table 4. Parameters for matrix clustering

execution mode	maximize density
area	100
density	70%
minimum rows/columns	10

6 Conclusion

We have explained the method for mining sequence patterns from WWW access log by using matrix clustering. Our method is superior to the conventional sequence mining algorithms with respect to the efficiency of execution, capability



A:492	F:854	K:835
B:742	G:846	L:827
C:493	H:834	M:862
D:4828	I:810	N:878
E:748	J:832	O:876

Fig. 11. Long sequence generated by the proposed method

to find a long sequence, and capability to find a generalized super-sequence from the resulting sequences. These superior attributes have been shown by the experiment using practical WWW access log. These characteristics are useful not only for analyzing WWW access log by end users, but also for personalization of WWW sites.

Next, we will discuss further research about this approach. In this article, problem definition is simplified by removing duplicated access of the same page. If we consider duplicated page access, the graph generation is not so simple because we must handle the multiple appearance of the same node in a graph. Further research is necessary for handling duplicated page access.

This article focused on the order of page access, but WWW log has other access information such as access time. Since the matrix clustering is based on the binary matrix of 2 dimension, it is difficult to handle various kinds of information such as access time. Further research is also necessary for handling various kinds of information and integrating various mining methods with matrix clustering.

References

1. M.J.A.Berry, G.Linoff : Data Mining Technologies: for marketing, sales, and customer support, John Wiley & Sons (1997)
2. Fukuda, Morimoto, Tokuyama: Data Mining, Kyoritsu-Pub. (2001) (in Japanese)
3. J.Han, L.V.S.Lakshmanan, J.Pei : Scalable Frequent-Pattern Mining Methods: An Overview, Tutorial Notes of KDD 2001 (2001)
4. R.Agrawal, R.Srikant : Fast Algorithms for Mining Association Rules, Proc. 20th VLDB Conf., (1994)
5. R.Agrawal, R.Srikant :Mining sequential patterns, Proc.Intl.Conf. Data Engineering (1995)
6. R.Srikant, R.Agrawal :Mining Sequential Patterns: Generalizations and Performance Improvements, Intl. Conf. Extending Database Technology(EDBT96), (1996)
7. H.Mannila, H.Toivonen, A.Verkamo: Discovery of frequent episodes in event sequences, Data Mining and Knowledge Discovery, Vol.1 pp. 259–289 (1997)

8. J.Han, J.Pei, M.Asl, Q.Chen, U.Dayal, M.Hsu : FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining, ACM Proc.KDD 2000,(2000)
9. J.Pei, J.Han, M.Asl, H.Pinto, Q.Chen, U.Dayal, M.Hsu : PrefixSpan: Mining Sequential Pattern Efficiently by Prefix-Projected Pattern Growth, Proc.2001 Intl. Conf. on Data Engineering(ICDE'01), (2001)
10. R.J.Bayardo.Jr : Efficiently Mining Long Patterns from Databases, ACM Proc.SIGMOD '98, (1998)
11. J.Srivastava, R.Cooley, M.Deshpande, P.Tan : Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data, SIGKDD Explorations, Vol.1,Issue 2(2000)
12. B.Mobasher, R.Cooley, J.Srivastava : Automatic Personalization Based on Web Usage Mining, Comm.ACM, Vol.43, No.8, pp. 142–151 (2000)
13. J.Schafer, J.Konstan, J.Riedl : E-Commerce Recommendation Applications, ACM Conference on EC,(2000)
14. B.Sarwar, G.Karypis, J.Konstan, J.Riedl : Analysis of Recommendation Algorithms for E-Commerce, ACM Conference on EC, (2000)
15. J.L.Herlocker, J.A.Konstan, Al Borchers and J.Riedl : An Algorithmic Framework for Performing Collaborative Filtering, Conf. Research and Development in Information Retrieval, (1999)
16. I.Dhillon : Co-clustering documents and words using bipartite spectral graph partitioning, KDD-2001, 269–274, ACM, (2001)
17. L.D.Baker and A.K.McCallum : Distributional clustering of words for text classification. ACM SIGIR Conference, (1998)
18. R.Kohavi : Mining E-Commerce Data: The Good, the Bad, and the Ugly, SIGKDD 2001 (2001)
19. S.Oyanagi, K.Kubota, A.Nakase: Matrix Clustering: A New Data Mining Method for CRM, Trans.IPSJ, Vol.42, No.8, pp. 2156–2166 (2001) (in Japanese)
20. S.Oyanagi, K.Kubota, A.Nakase : Application of Matrix Clustering to Web Log Analysis and Access Prediction, WEBKDD 2001 (2001)

Comparing Two Recommender Algorithms with the Help of Recommendations by Peers

Andreas Geyer-Schulz[1] and Michael Hahsler[2]

[1] Universität Karlsruhe (TH), D-76128 Karlsruhe, Germany,
Andreas.Geyer-Schulz@em.uni-karlsruhe.de
[2] Wirtschaftsuniversität Wien, A-1090 Wien, Austria,
Michael.Hahsler@wu-wien.ac.at

Abstract. Since more and more Web sites, especially sites of retailers, offer automatic recommendation services using Web usage mining, evaluation of recommender algorithms has become increasingly important. In this paper we present a framework for the evaluation of different aspects of recommender systems based on the process of discovering knowledge in databases introduced by Fayyad et al. and we summarize research already done in this area. One aspect identified in the presented evaluation framework is widely neglected when dealing with recommender algorithms. This aspect is to evaluate how useful patterns extracted by recommender algorithms are to support the social process of recommending products to others, a process normally driven by recommendations by peers or experts. To fill this gap for recommender algorithms based on frequent itemsets extracted from usage data we evaluate the usefulness of two algorithms. The first recommender algorithm uses association rules, and the other algorithm is based on the repeat-buying theory known from marketing research. We use 6 months of usage data from an educational Internet information broker and compare useful recommendations identified by users from the target group of the broker (peers) with the recommendations produced by the algorithms. The results of the evaluation presented in this paper suggest that frequent itemsets from usage histories match the concept of useful recommendations expressed by peers with satisfactory accuracy (higher than 70%) and precision (between 60% and 90%). Also the evaluation suggests that both algorithms studied in the paper perform similar on real-world data if they are tuned properly.

1 Introduction

Since recommender systems are becoming widely used by retailer Web sites (e.g. Amazon.com, Barnes & Noble.com), a careful evaluation of their performance gets increasingly important. However, recommender systems are complex applications that are based on a combination of several models (mathematical and psychological), algorithms, and heuristics. This complexity makes evaluation very difficult and results are hardly generalizable, which is apparent in the literature about recommender systems (for a survey see table 2 in this paper).

In this paper we try to improve the evaluation of recommender systems by developing a more systematic approach in respect of what actually is evaluated. For this purpose we develop in section 2 a framework for the systematic evaluation of recommender

O.R. Zaïane et al. (Eds.): WEBKDD 2002, LNAI 2703, pp. 137–158, 2003.

systems which is in principle based on the process of knowledge discovery in databases by Fayyad et al. [1]. The expected benefit of this is that the evaluation of various aspects of a recommender system can be separated and thus more properly targeted by the different stakeholders responsible for the introduction of such a system. Therefore, our framework is more suitable for continuous process improvement by focusing on the most promising areas of improvement. In section 3 we review common performance measures for recommender system evaluation and how they are used in the recommender systems literature.

In the framework in section 2 we identified a performance evaluation method which evaluates how well the interpretation of patterns extracted by recommender algorithms matches the concept of useful recommendations given by a human peer. Although, Resnick and Varian [2] define recommender systems explicitly as systems supporting the social process of recommending products to others, this question is often neglected in the literature of recommender systems based on usage mining. To fill this gap we apply a performance evaluation method to compare two data mining methods for the generation of recommendations with recommendations by peers.

There are many possible sources for generating recommendations including expressed preferences, opinions of experts, characteristics of people and items, and many more. For recommendation services the information of all available sources should be combined to provide the best recommendations possible. However, for the performance evaluation of usage mining algorithms in this paper we restrict our recommendation generation to the case where the only available information is a collection of past usage data. The active user is anonymous and we only know the last item the user chose. This seems to be very restrictive. However, Web retailers and Internet information providers have to deal with such a situation every day since the majority of users browse the Web most of the time anonymously and still services like recommendations right after the first click are needed to improve sales. In sections 4 and 5 we give a brief introduction to the two data mining algorithms used in the paper. The first algorithm uses the in the knowledge discovery in databases (KDD) society well-known support-confidence framework, the second algorithm is based on Ehrenberg's repeat-buying theory originating from marketing research. In section 6 we describe the experimental setup and the data set. In section 7 we present and discuss the evaluation results. We conclude with a short summary of the findings and open research questions in section 8.

2 A Framework for the Evaluation of Recommender Systems

Recommender systems are complex applications which have to perform the whole process of knowledge discovery in databases (KDD process). In [1] Fayyad et al. give an overview of the steps of the KDD process. An application of this model to recommender systems is straightforward. In order to separate the influence of the user interface from the effects of the choice of a data mining method we add *presentation* as additional step to the process model of Fayyad et al. (see figure 1). The five major steps are:

1. *Selection:* The data set used to produce recommendations can stem from various sources. For example these sources can be already existing transaction logs (e.g.

point of sale data, Web server logs) or the data can be collected especially for the purpose of generating recommendations (e.g. ratings for movies).

2. *Preprocessing and transformation:* In these steps the data set is cleaned from noise, inconsistent data is removed, and missing data is inferred. After this treatment the cleaned data is transformed into a representation suitable for data mining.
   For example, for collaborative filtering the data normally consists of explicit ratings by users which are collected for the purpose of creating recommendations (e.g. for music see [3]). Preparation mainly involves discovering and removing inconsistent ratings.
   For Web usage mining (see [4,5]) data is collected by observing the behavior of users browsing a Web site. Since observation, especially server-side observation on the Web, is far from perfect, much effort has to be put on data preprocessing, cleaning and transformation. Problems involve the identification of users, dynamic (rotating) IP-addressing, session identification, missing data due to proxy servers and system crashes, requests by Web robots – to name just a few.
3. *Data mining:* The objective of this step is to find interesting patterns in the data set that are useful for recommendation purposes. The output of data mining in recommender systems can be: groups of users with similar interests, items that are frequently used together, often used sequences of items,... Frequently, extracting patterns means learning the parameters of a specified model from the data set.
4. *Interpretation and evaluation:* In order to build knowledge, the found patterns (the model and its parameters) have to be understandable to humans. Only with this property the process can be called knowledge discovery and the results can be interpreted. A recommender system interprets found patterns for the user.
   Finally the validity (patterns are also valid for new data), novelty (involves a new way of finding patterns), usefulness (potentially lead to useful action), and understandability (build and increase knowledge) of the patterns need to be evaluated.
5. *Presentation:* A recommender system presents this interpretation in a suitable form as a recommendation. There are many presentation options. For example, the recommendation can be a top-n list of recommended items for a user, or a list of items that are similar to an item the user likes, or it can consist of information about how other users with similar interests rated a specific item.

Wirth and Hipp included in their process model for data mining called CRISP-DM [6] a step called deployment. In this step they emphasize the need to deliver the results or even a ready-to-use implementation of the data mining process to the customer. Their definition is similar to software engineering, where the deployment phase usually includes the actual introduction and operation of a system in an organization. Developing an evaluation methodology for the deployment phase of a recommender system is beyond the scope of this paper.

Because of the complexity of recommender systems, and the many available choices for each step of the knowledge discovery process described above, detailed evaluation of the different aspects of this process is a necessity. In figure 1 we mapped five evaluation methods for recommender systems to the steps of the KDD process to provide a systematic reference framework. We summarize these evaluation methods in the following:

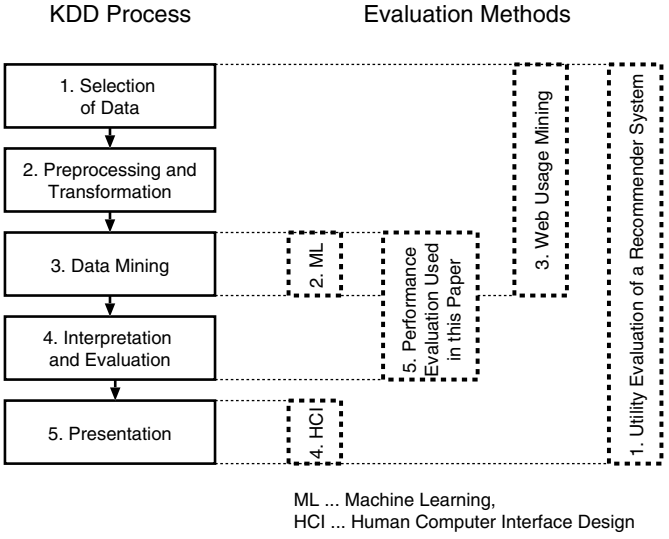


Fig. 1. Mapping evaluation methods to the steps of the KDD process

1. The evaluation of the utility of a recommender system is targeted to the stakeholders of the organization. The evaluation spans the whole process and assesses the utility of process as a whole as indicated by the right-most evaluation process labeled "1. Utility Evaluation of a Recommender System" in figure 1. In practice the utility of the whole recommender system is usually evaluated by the impact of the recommender system on the overall performance of the process the recommender supports. If the process supported is the purchase process in a supermarket, the impact is measured by comparing sales in two test markets, one with and one without the recommender system. This approach is used in Lawrence et al. [7] for the evaluation of a personalized product recommender on personal digital assistants (PDAs) for the Safeway supermarket chain. The performance measure used for evaluation is the revenue from the sales volume triggered by the recommender. Lawrence et al. reported that they observed an increase in revenue of about 1.8% after introducing the recommender to a new store. This is in line with NetPerception's estimation of a 2.0% increase in sales volume [8]. NetPerception measures this by comparing the impact of random product lists versus recommended product lists on the buying behavior of the consumer. For non-profit sites, such an overall performance measure could be a conversion rate, a contact rate or a task achievement rate.
2. Most researchers in recommender systems focus on the evaluation of mining algorithms with methods known from *machine learning.* A common way from machine learning for comparing the performance of several algorithms is to use a prepared data set and divide it into a set for training and a set for evaluation (cross-validation is often used). This approach only takes into account how well patterns in the data set can be learned and not how useful the patterns are for recommendation purposes. However, in addition, the underlying theoretical framework of these algorithms must

be evaluated with respect to its consistency. For association rule algorithms, a recent discussion can be found in Adamo [9, pp. 151-184]. For collaborative filtering based on explicit product ratings a comparison with regression models and a survey of studies in this area can be found in [10]. Furthermore, for model-based approaches the correspondence of the model with reality must be evaluated. For statistical models this is done by testing the underlying behavioral assumptions in a piece-meal fashion, by diagnostic-checking. For the repeat-buying theory used below in this paper, see e.g. [11]. Model comparison is performed along several dimensions, most notably performance, robustness, parsimonity of parameters, sensitivity to misspecification of parameters, computational complexity, ease of use and understandability.

3. *Web usage mining* covers the first three steps of the KDD process. In addition to the evaluation of the data mining method evaluation of the data selection and preprocessing steps is important for Web usage mining [4,5]. Correctly observing user behavior on the Web requires automatically handling difficulties e.g. with filtering automatic Web robots [12], detecting sessions [13] and path completion [14]. For example, for evaluating session identification heuristics Cooley recommends comparing the results of log files with session identification (e.g. by cookies or by server-side session identifiers embedded in the link) with the results of analyzing the same log files stripped (see [14, pp. 118-123]). Another evaluation method is testing preprocessing statistics on synthetic data. For episode identification this approach has been used by Cooley [14]. Berendt et al. [15] define quality measures for session reconstruction from log files and compare different heuristics based on session duration, time spent on a page and referrer values. In their analysis they found considerable performance differences between the heuristics depending on the structure of the Web site.
   However, while these studies ([14] and [15]) make a considerable contribution to the evaluation of Web usage mining, they are still based on real data sets. The evaluation of the quality of preprocessing heuristics is still biased from measurement errors (e.g. unidentified robots, ...) and on assumptions on actual use and spread of technology (e.g. dynamic IP addresses). Therefore, the construction of synthetic data sets with known properties (or the use of fully instrumented Web sites and browsers in a controlled environment) and subsequent masking of information as evaluation suites for Web usage mining methods would be of considerable value. With the coming of ad-hoc networking and a considerable growth in mobile, embedded and wearable devices progress in this direction is necessary.
4. Evaluation of the presentation of recommendations to the consumer/user is dealt with in the area of *human-computer interface (HCI)* research and includes methods such as usability labs and field-experiments. In connection with recommender systems Herlocker et al. [16] compared 21 different representations of recommendations for movies. The findings were that – similar to previous experiences with expert systems – suitable explanation of the recommendations to the users increases the acceptance of the recommender system.
5. *The performance evaluation of recommender systems used in this paper* evaluates how well the interpretation of patterns extracted by a recommender algorithm matches the concept of useful recommendations given by a peer. This evaluation combines the data mining step as well as part of the interpretation step of the KDD

Table 1. 2x2 confusion matrix

actual / predicted	negative	positive
negative	a	b
positive	c	d

process shown in figure 1. Since such a performance evaluation is not common in the very machine learning oriented recommender literature, we provide a evaluation of a simple association rule algorithm and a novel repeat-buying based algorithm in this paper.

3 Performance Measures for Recommendation Algorithms

For measuring the performance of recommender algorithms measures originating from statistics, machine learning, and information retrieval are used. To give definitions of the performance measures we first have to define the meaning of the terms item, recommendation, and recommendation list. *Items* are the products under consideration that can be purchased (consumed/used) by the customer (user). A *recommendation list* is a list of items which the algorithm recommends for a user in a certain situation. We call each item in the list a *recommendation* which can be either correct or incorrect.

Since all measures use similar information it is useful to define them with the help of the so called *confusion matrix* depicted in table 1 (see [17]) which corresponds exactly to the outcomes of a classical statistical experiment. The confusion matrix shows how many of the possible recommendations were predicted as recommendations by the algorithm (column predicted positive) and how many of those actually were correct recommendations (cell d) and how many not (cell b). The matrix also shows how many of the possible recommendations the algorithm rejected (column predicted negative), were correctly rejected (cell a) or should have actually been recommended (cell c). Statistically we test the hypothesis H_0 that an item is a recommendation (positive in table 1) against the hypothesis H_1 that an item is not a recommendation (negative in table 1). Cell c is known as type I error with probability α and cell b is known as type II error with probability β.

Performance Measures from Machine Learning. For the data mining task of a recommender system the performance of an algorithm depends on its ability to learn significant patterns in the data set. Performance measures used to evaluate these algorithms have their root in machine learning.

Commonly used measures are accuracy and coverage. *Accuracy* is the fraction of correct recommendations to total possible recommendations (see formula 1). *Coverage* measures the fraction of items the system is able to provide recommendations for (see formula 2). We can not define coverage directly from the confusion matrix, since the confusion matrix only represents information at the level of recommendations (relationships between items) and not at the level of individual items with recommendation lists.

$$Accuracy = \frac{correct\ recommendations}{total\ possible\ recommendations} = \frac{a+d}{a+b+c+d} \tag{1}$$

$$Coverage = \frac{items\ with\ recommendations}{total\ number\ of\ items} \tag{2}$$

A common error measure is the *mean absolute error (MAE, also called mean absolute deviation MAD)* shown in formula 3. N is the length of the learned pattern from the training set (the total number of items for which recommendations are produced = items with recommendations in formula 2) and $|\epsilon_i|$ is the absolute error of each component (number of incorrect classifications in the recommendation list for each item) of the pattern compared to the evaluation set.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\epsilon_i| = \frac{b+c}{N} \tag{3}$$

Performance Measures from Information Retrieval. Recommender systems help to find items of interest from the set of all available items. This can be seen as a retrieval task known from information retrieval. Therefore, standard information retrieval performance measures are frequently used to evaluate recommender performance.

Precision and *recall* are the best known measures used in information retrieval [18, 19] (see formula 4 and 5 for the definitions).

$$Precision = \frac{correctly\ recommended\ items}{total\ recommended\ items} = \frac{d}{b+d} \tag{4}$$

$$Recall = \frac{correctly\ recommended\ items}{total\ useful\ recommendations} = \frac{d}{c+d} \tag{5}$$

Often the number of *total useful recommendations* needed for recall is unknown since the whole collection would have to be inspected. However, instead of the actual *total useful recommendations* often the total number of known useful recommendations is used as an estimate.

Precision and recall are conflicting properties, high precision means low recall and vice versa. To find an optimal trade-off between precision and recall a single-valued measure like the *E-measure* [19] can be used. The E-measure is defined in formula 6. The parameter α controls the trade-off between precision and recall.

$$\textit{E-measure} = \frac{1}{\alpha(1/Precision) + (1-\alpha)(1/Recall)} \tag{6}$$

A popular single-valued measure is the *F-measure.* It is defined as the harmonic mean of precision and recall given in formula 7. It is a special case of the E-measure with $\alpha = .5$ which places the same weight on both, precision and recall. In the recommender evaluation literature the F-measure is often referred to as the measure *F1*.

$$\textit{F-measure} = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2}{1/Precision + 1/Recall} \tag{7}$$

Other Performance Measures Used for Recommender Systems. Other measures are often model dependent like *r-square* and *root mean squared error (RMSE)* for regression models using ratings. Various *hit rates* are also possible, comparing user ratings with the computed recommendations.

A measure used in the literature to compare two algorithms or parameter settings is the *Receiver Operating Characteristic* (ROC). It is a measure used in signal detection and goes back to the Swets model [19]. The ROC-curve is a plot of the system's *probability of detection* (also called $sensitivity$ or true positive rate or recall as defined in formula 5) by the *probability of false alarm* ($1 - specificity$, where $specificity = \frac{a}{a+b}$ which is the true negative rate) with regard to model parameters. A possible way to compare the efficiency of two systems is by comparing the size of the area under the ROC-curve, where a bigger area indicates better performance.

In the context of real-time personalization for Web sites Mobasher et al. introduced in [20] variants of performance measures which evaluate a recommendation list R with regard to a single user session t ("a transaction") and the window $w \subseteq t$ used for production of the recommendation list R. $\mid t \mid$, $\mid w \mid$, and $\mid R \mid$ denote the sizes of these sets. The variants for precision and coverage are:

$$Precision(R,t) = \frac{\mid R \cap (t-w) \mid}{\mid R \mid} \qquad Coverage(R,t) = \frac{\mid R \cap (t-w) \mid}{\mid t-w \mid} \tag{8}$$

In addition, they propose the R-measure (coverage divided by the size of R) which favors smaller recommendation lists.

In table 2 we summarize recent papers that evaluated recommender algorithms using the presented measures.

4 A Simple Recommender Using Association Rules

The first recommender we use is based on association rules with thresholds for minimum support and minimum confidence. This is known as the support-confidence framework. The problem of finding association rules for market basket data that satisfy minimum support and minimum confidence was first presented by Agrawal et al. [27]. Association rule algorithms require no model assumptions. This makes the approach very attractive and simple because checking whether the model assumptions are met is not required.

The problem is formalized in [27] as follows: Let $I = \{i_1, i_2, ..., i_n\}$ be a set of items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. A transaction T contains X if $X \subseteq T$ and Y if $Y \subseteq T$. An association rule is an implication in the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and X, Y disjoint. X is called the antecedent and Y is called the consequent of the rule.

For association rules the thresholds of the two measures – minimum support and minimum confidence – are commonly used to identify a subset of rules for detailed analysis or as input for a recommender application. Support is a measure of statistical significance. The rule $X \Rightarrow Y$ has support sup in the transaction set D if more than $sup\%$ of transactions in D contain X and Y together. Confidence is a measure of strength. The rule $X \Rightarrow Y$ holds in the transaction set D with confidence $conf$ if $conf\%$ of

Table 2. Recommender algorithm evaluation papers

Paper	Domain	Algorithm	Measures
Shardanand and Maes [3]	Music ratings	Prediction algorithms based on similarity between user profiles	MAE
Herlocker et al. [21]	Movie ratings	Neighborhood based prediction algorithms	MAE, ROC, Coverage
Sarwar et al. [22]	Movie ratings, E-Commerce purchases	Dimensionality reduction	MAE, F-measure
Mobasher et al. [23]	Web site usage	Aggregate user profiles (clustering user transactions and pageviews)	Accuracy
Vucetic and Obradovic [24]	Movie ratings	Regression based item-to-item relationship	MAE, ROC
Yu et al. [25]	Movie ratings	Instance selection for collaborative filtering	MAE, Accuracy
Mobasher et al. [20]	Web site usage	Aggregate user profiles (clustering user transactions and pageviews)	Precision, Coverage, F-measure, R
Lin et al. [26]	Movie ratings	Adaptive-support association rule mining	Accuracy, Precision, Recall
Mild and Natter [10]	Movie ratings	Collaborative filtering, linear regression models e.g. with model selection	MAE, RMSE, R-square, hit-rate

transactions in D that contain X also contain Y. In formula 9 and formula 10 support and confidence are defined by probabilities.

$$sup(X \Rightarrow Y) = sup(Y \Rightarrow X) = P(X \wedge Y) \tag{9}$$

$$conf(X \Rightarrow Y) = \frac{P(X \wedge Y)}{P(X)} = \frac{sup(X \Rightarrow Y)}{sup(X)} \tag{10}$$

Agrawal and Srikant presented in [28] the APRIORI algorithm to efficiently compute association rules with several items in the antecedent of the rule using frequent itemsets. A frequent itemset is a set of items that satisfy minimum support in the set of transactions. The algorithm generates larger frequent itemsets with every pass by combining frequent itemsets from the last pass and pruning away the itemsets without sufficient support. The algorithm stops when no larger itemset can be produced without falling under minimum support. Then all large itemsets (frequent itemsets that could not be combined to larger frequent itemsets) are used to produce the association rules.

For the simple recommender system used in this paper we only need association rules with one single item in the antecedent of the rule. Then for the item in the antecedent of each rule we use the items in the consequent as the recommendation list. The number of recommendations in the list can be varied by changing the support and confidence thresholds.

Recently support and confidence were subject to criticism [29,30]. The main point of criticism is that confidence ignores the frequency of the consequent in the data set and therefore spurious rules with items in the consequent that appear often in the data

set cannot be separated from more accurate rules. *Lift* [31], also known as *interest* [29, 30] is an alternative measure for confidence that takes the frequency of the consequent into account. But in contrast to implication it measures only the co-occurrence of X and Y as a symmetric measure. Lift is defined as the relation of the (observed) probability of the co-occurrence of two items to the probability under the assumption that they occur independently. The definition of lift for the rules $X \Rightarrow Y$ and $Y \Rightarrow X$ is given in formula 11.

$$lift(X \Rightarrow Y) = lift(Y \Rightarrow X) = \frac{P(X \wedge Y)}{P(X)P(Y)} = \frac{conf(Y \Rightarrow X)}{sup(X)} \tag{11}$$

Another alternative measure for confidence is *conviction* (see [29] and [31]). Conviction measures the deviation of the implication $X \Rightarrow Y$ from the assumption that X and Y occur independently. It is derived from the fact that the implication $X \Rightarrow Y$ can logically be reformulated as $\neg(X \wedge \neg Y)$. If we divide this by the individual probabilities $P(X)$ and $P(\neg Y)$ and invert the ratio to compensate for the outside negation we reach the definition of conviction as given in formula 12.

$$conviction(X \Rightarrow Y) = \frac{P(X)P(\neg Y)}{P(X \wedge \neg Y)} = \frac{1 - sup(Y)}{1 - conf(X \Rightarrow Y)} \tag{12}$$

Although in the literature conviction is claimed to be superior to confidence [29] most papers still use the support-confidence framework. For an extensive survey of variants of association rule algorithms and a discussion of their intrinsic short-comings, see Adamo [9]. We will report results for confidence, lift, and conviction in this paper.

5 A Simple Recommender Using the Repeat-Buying Theory

The second recommender algorithm is based on the repeat-buying theory introduced by Ehrenberg [11]. In the context of this paper, buying an item means to visit a Web site. The idea behind the repeat-buying theory is that in a stationary situation with the purchases of all items being independent from each other the buying behavior follows a process that produces a specific distribution for the number of repeat buys, namely the negative binomial distribution (NBD). The statistical model is based on several strong behavioral assumptions about consumer purchase behavior. Although these assumptions invite criticism of the model, Ehrenberg [11] and other authors empirically showed that this model holds for various consumer markets.

In [32] we showed how to apply a simplified form of the model (using the logarithmic series distribution (LSD), a limit case of the NBD also described in [11]) to generate recommendations for the Internet information broker used in this paper. In this setting the LSD in formula 13 gives the probability of the observation that the same pair of two independent items are used together in a specified period of time a total of 1, 2, 3, ..., r times by pure chance. q is the only parameter of the distribution and can be easily estimated. First we calculate the mean of the observed usage frequency w for all item pairs that contain one specific item. And then we approximate q from w by their relationship stated in formula 14.

Table 3. Algorithm for computing recommendations based on repeat-buying

Algorithm GenerateRecommenderLists

for each item x in all transactions **do** {

1. generate the frequency distribution of co-occurrences from all transactions containing the item x
2. approximate the (only) parameter q of the LSD distribution from the mean w of the frequency distribution
3. generate an empty recommendation list for item x
4. **while** the expected type II error rate β is below a set threshold **do** {
   a) add the item with the highest number of co-occurrence to the list of recommendations
   b) compute the expected type II error rate for the new list }
5. store the recommendation list }

$$P(r) = \frac{-q^r}{r \ln(1-q)}, \quad r \geq 1 \tag{13}$$

$$w = \frac{-q}{(1-q)\ln(1-q)} \tag{14}$$

In [32] we showed that the LSD model can be fitted to co-occurrences of information products in our information broker reasonably well. However, the LSD model is only applicable to co-occurrence of items under the strict assumption of independence between the usage of all items. Of course, this assumption is not valid for some items in the data, caused by dependencies between the usage of items that cover the same topic or that are useful as complementary information sources. For the recommender system we need to find the items with dependencies. Since the dependencies between items violate the strict independence assumption of the model outliers that do not follow the fitted LSD are produced. For each pair of items we can calculate the probability that it comes from the LSD model by dividing the observed number of co-occurrences by the predicted number (using $P(r)$ from formula 13). This results is an estimate of the type II error for each possible recommendation.

The algorithm in table 3 produces for each item x a recommendation list from the co-occurrences with different y_i that has an expected type II error rate (β) below a predefined threshold. By changing this threshold, recommendation lists with higher or lower expected β and potentially more or less recommendations can be produced by the algorithm.

The algorithm produces similar results as association rules using variable confidence and support thresholds. If we set support to 0 and we order all association rules with the same antecedent x by confidence we get the same set of co-occurrences that is used to calculate the LSD. Since confidence preserves the rank order of the number of co-occurrences with the same antecedent the selection of recommended items is made in the same sequence for both algorithms. The difference between the algorithms is the way how the threshold for the recommended items is calculated. For association

rules recommendations consist of rules that satisfy a predefined minimum support and confidence threshold, where support is calculated over all transactions. An exception to this is the approach of Lin et al. [26] which requires the specification of a target range for the number of recommendations offered to a specific user and which adapts the support threshold so that a rule set with the desired size is generated. For the repeat-buying algorithm the set of recommendations is selected for each item x individually from the frequency distribution of co-occurrences, so that the total expected type II error rate (in comparison with the fitted model) is below a set threshold. In association rule terminology this procedure amounts to automatically finding for all rules with the same antecedent an individual confidence threshold that satisfies the error constraint.

Association rule algorithms require the computation of support and confidence (or lift or conviction) and the specification of two threshold parameters for these. The computational complexity is in general of exponential order. For the simple association rules with only one item in the antecedent used in this paper, the computational complexity is of quadratic order.

Production recommender systems in organizations require periodical updates. For association rule algorithms updating techniques which consider the effects of the update on all current association rules are presented in [33], [29], and [34]. Cheung et al. [33] design an updating algorithm which exploits necessary conditions on the support of an itemset in the update increment to be a large itemset to efficiently identify winners (itemsets that become large itemsets after the update) and loosers (itemsets which cannot become large itemsets after the update) and thus realize considerable improvements in efficiency. Nevertheless, all support and confidence values of all association rules must be recomputed. Brin et al. [29] discuss the handling of incremental updates for finding large itemsets as a potential, but straightforward extension of the DIC algorithm. Unfortunately, however, this helps only in computing large itemsets, support and conviction have to be recomputed for all association rules. Ng and Lam [34] improve the work of Cheung et al. and Brin et al. with the help of sophisticated dynamic itemset counting techniques. However, similar to the other approaches, all effort is invested in identifying large itemsets, all support and confidence values must be recomputed. Therefore, the update of the confidence and support of all current association rules requires a complete update of all support and confidence values which is of quadratic order in the number of items.

Like the simple association rule algorithm used in this paper the computational complexity of the repeat-buying algorithm is $O(n^2)$, where n is the number of items. However, an incremental update version of the algorithm is easy to implement. The algorithm simply requires an update of the count of all co-occurrences in the increment. Only for actually updated items LSD-models must be recomputed and the complexity is reduced to $O(n_u^2)$ with n_u the number of items updated. The reason for this is that the theoretical LSD depends only on the sample mean of all co-occurrence counts with the same item (see equation 14) and not on any global parameter of the dataset. This is an important advantage over simple association rule algorithms which gets more pronounced as the ratio of the total number of items to the number of items used between two updates increases. A real-world example where we benefit from this difference is

computing recommendations for the users of a large research library's online public access catalog (OPAC) with millions of items and $n_u \ll n$.

6 Experimental Setup

In this section we compare the performance of the two recommender algorithms for the data mining step as well as for the interpretation and evaluation step of the KDD process. Most papers (except [7] and [8]) only evaluate the data mining step using techniques from machine learning. The evaluation methods in [7] and [8] are not applicable for non-profit organizations as e.g. universities, research libraries, and government agencies because of missing data on sales volume or profits. However, Resnick and Varian's seminal paper [2] defined recommender systems as systems supporting the social process of recommending products to others – as systems supporting the word-of-mouth effect well known in marketing. Having this definition in mind, it is a natural question if the recommendations produced by a system are similar to the recommendation produced by a peer (a person in the target group of the system). Therefore, we asked for each item in a recommendation list the question if the interviewed person would recommend it to a person interested in the item the list was produced for. The advantage of this approach is that it does not require the setup of test markets (which are quite costly to run) and that it can easily be applied in non-profit organizations, too. In addition, by judiciously controlling the subjects in the sample – a sort of control not yet sufficiently exercised in this paper – tuning recommender systems to heterogenous target groups may be improved.

For the study we use a data set from the Virtual University information system at the Wirtschaftsuniversität Wien. This information system is an educational Internet information broker that provides access to online information products (e.g. lecture notes, research material and enrollment information) for students and researchers. An information product in this setting is defined as a set of Web pages (including other media like word processor files) that present a logical unit of information on a specific topic, e.g. a Web site on software development. The information products are distributed all over the Internet and the information broker logs the mediations of the products at the application level. The transaction data is anonymous but includes a cookie based session management and Web robots are removed.

To generate recommendations for the information system we provide the recommender algorithms with a data set containing market basket data obtained from the log of the information broker. A market basket is the set of all items (products) used together in a single transaction (a session). We use 6 months of usage data (January to June 2001) containing 25522 transactions with more than one item. These transactions contain 3774 different items with 114128 co-occurrences of items that are possible recommendations.

For evaluation we produced for each item a list of all other items that co-occurred together with the first item in at least one transactions and randomly selected 300 lists. For each selected list we produced a computer-based questionnaire with the list of items in randomized order. The items were represented in the questionnaire with their names and we also provide the links to the corresponding Web pages so the interviewed person could browse through the actual information. For each item we asked the question

if the interviewed person would recommend it to a person interested in the item the list was produced for. The possible answers were "recommend" or "don't recommend" which correspond to the perceived usefulness of a possible recommendation. We asked people from the information system's target group (6 students, 1 system administrator, 1 secretary, and 5 researchers from the Universität Karlsruhe (TH)) to evaluate each an exclusive subset of the lists. The number of possible recommendations in each subset was approximately equal, so that no single subject had a dominating influence on the evaluation set. Exclusive subsets implies that inter-rater reliability can – unfortunately – not be assessed and compared with the performance of the algorithms. In addition, subjects were not randomly selected. However, any uncontrolled effects because of the missing randomization of the subjects applies to both algorithms in the same way.

In total 1661 possible recommendations (co-occurrences of items) were evaluated. For 561 co-occurrences the interviewed persons said they would recommend the latter item, and for 1100 they would not. More than 120 lists are of the size 2 (the smallest size analyzed) and only few lists have a size larger than 50. This is in line with the characteristics of other real-world datasets used in Zheng et al. [35]. After visual inspection the proportion of good items in the lists seems to be independent from the size of the list with an average of 0.338.

7 Evaluation Results

In this section we present and discuss the evaluation results for the different recommender algorithms. The algorithms used the data set described above as input and the produced recommendations were evaluated using the useful recommendations identified by the users.

To find sensible support values for the association rules based algorithms, we varied minimum confidence between 0.3 and 0.000001 (0.3, 0.2, 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001) and plotted precision by recall for several support thresholds. Figure 2 depicts the precision/recall plots for the best performing support thresholds. In the classic setting for association rules, e.g. in a supermarket setting, the analyst is interested in a manageable, very small set of rules with high support. In this setting high support means to only consider products with high sales and therefore to justify the effort to physically rearrange the products in the market. However, for an association rule algorithm producing only a small set of rules means low recall and low coverage. In an on-line recommender system there is no (or very small) additional cost for displaying more (useful) recommendations in the interface. Therefore, we can also efficiently present recommendations for products with low support – as long as those recommendations are useful. To obtain reasonable recall for our data set, the minimum support threshold has to be chosen relatively small with values between 0.001 and 0.00001 (see figure 2). This is due to relatively short transactions and the high number of items in the data set.

Next we compare the performance of confidence with the alternative measures of interestingness, lift, and conviction. For recommender systems recommendation lists with low precision are problematic since recommending unrelated items annoys the users. Since reasonable precision starts with 0.5 and up, we use only minimum support of 0.0001 and 0.00003 (see figure 2) for the comparison. Figure 3 shows the precision/recall plots

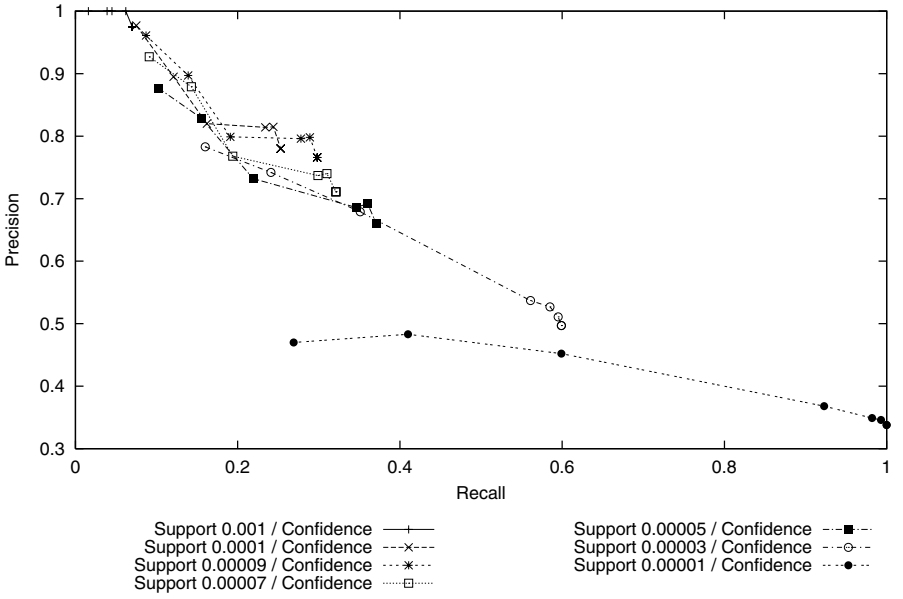


Fig. 2. Precision/recall plot for association rules with different minimum support

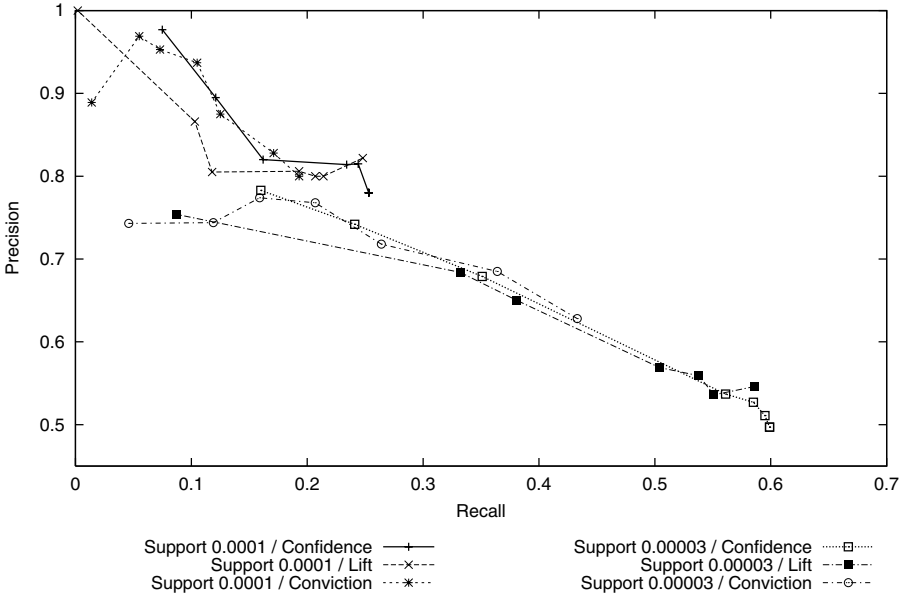


Fig. 3. Precision/recall plots for confidence, lift, and conviction

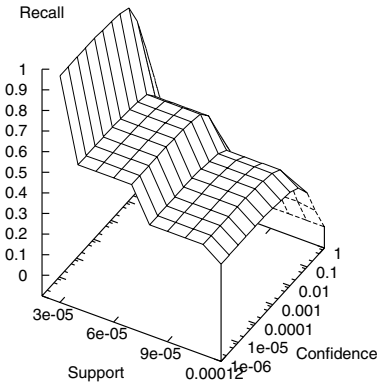


Fig. 4. Recall for association rules by support and confidence threshold

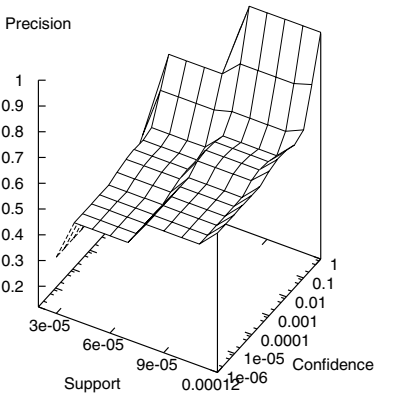


Fig. 5. Precision for association rules by support and confidence threshold

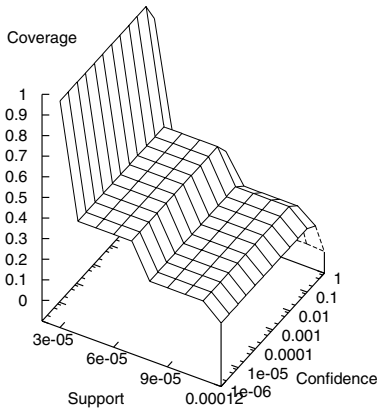


Fig. 6. Coverage for association rules by support and confidence threshold

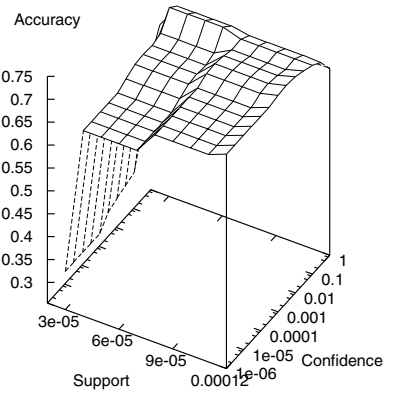


Fig. 7. Accuracy for association rules by support and confidence threshold

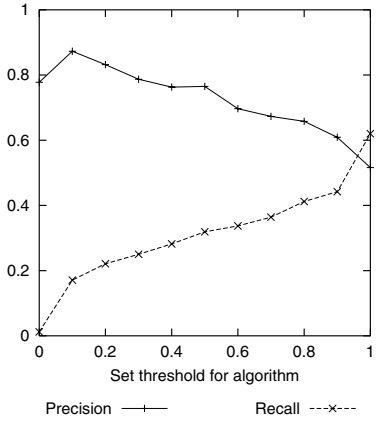


Fig. 8. Precision and recall by the threshold of the repeat-buying algorithm

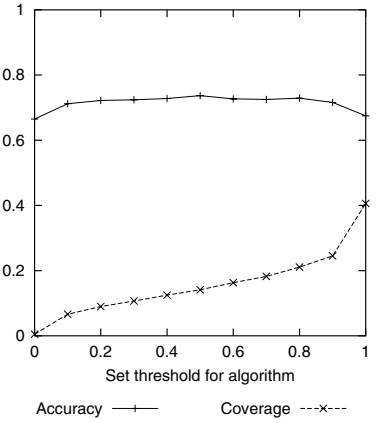


Fig. 9. Accuracy and coverage by the threshold of the repeat-buying algorithm

for the two selected minimum supports. For the plots we varied lift between 1.5 and 1000 and conviction between 1.05 and 2. Since neither lift nor conviction perform significantly better than confidence on the data set we use the support-confidence framework for all further investigations.

Figures 4 to 7 show the surface plots of recall, precision, coverage, and accuracy by minimum support and confidence (confidence is plotted on a logarithmic scale). Note, that these measures must be optimized simultaneously. Choosing a certain combination of support and confidence determines all four measures simultaneously, not every combination of performance levels (e.g. a recall larger than 0.9 and a precision larger than 0.9) can be attained by the algorithm. As expected, recall (see figure 4) and coverage (figure 6) decrease with increasing minimum support, but stay almost constant for most of the plotted range of minimum confidence. Only for confidence values bigger than 0.01 (0.1 for coverage) the measures decrease fast. With support the measures decrease in several steps, indicating that many rules have the same level of support. Precision (see figure 5) and accuracy (figure 7) show a less stable behavior. Precision decreases with smaller minimum support and minimum confidence and strives to 1 for very high minimum confidence (producing very few rules). However, there is unevenness at the same level of support which is also visible in the accuracy plot. Precision and accuracy deteriorate fast for very small values of minimum support (< 0.00003). Between minimum support of 0.00006 and 0.00007 appears an abrupt step, indicating that below this level many association rules with a high error rate are accepted by the algorithm. Minimum confidence has less influence on accuracy than minimum support, however, the confidence level for maximum accuracy changes with the set minimum support between 0.03 for higher support (> 0.00007) to 0.3 for lower support (< 0.00004), indicating that there is a strong interdependence between both parameters and the structure of the data set.

Table 4. Results of the repeat-buying algorithm for different threshold values

Threshold	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Precision	0.78	0.87	0.83	0.79	0.76	0.77	0.7	0.67	0.66	0.61	0.52
Recall	0.01	0.17	0.22	0.25	0.28	0.32	0.34	0.36	0.41	0.44	0.62
Accuracy	0.67	0.71	0.72	0.72	0.73	0.74	0.73	0.73	0.73	0.72	0.68
Coverage	0.02	0.13	0.17	0.19	0.22	0.23	0.25	0.31	0.38	0.42	0.44
Avg. type II error rate	0.17	0.13	0.22	0.23	0.26	0.26	0.3	0.33	0.33	0.36	0.46

For the repeat-buying algorithm we varied the threshold between 0 and 1 and calculated the same quality measures as above. Table 4 and figures 8 and 9 contain the results. With a higher threshold the algorithm becomes less selective and more recommendations are produced. Therefore, recall and coverage are rising with an increasing threshold and precision is decreasing. With only a few exceptions (for small threshold values, where only few recommendations are generated) these quality measures change monotonously with the threshold. Accuracy is almost constant at a level around 0.7 which means that with higher thresholds the type I error decreases at a similar rate as the type II error increases.

In figures 10 and 11 we compare the performance of the association rule algorithm and the repeat-buying algorithm in terms of precision by recall and accuracy by coverage. For comparison, we included in both plots a recommender that chooses recommendations randomly from the co-occurrence list with the probability varying between 0 and 1. Both recommender algorithms perform significantly better in predicting the items that users qualify as useful recommendations than choosing recommendations randomly. The repeat-buying recommender performs similar to the association rule algorithm with the support-confidence framework. Figure 10 shows that at reasonable recall (between 20% and 50%) both algorithms reach a precision between 60% and 80% which is acceptable for most applications. Both algorithms provide accuracy above 70% (see figure 11), however, the support-confidence framework is very brittle with respect to changes of minimum confidence and, what is more problematic, the optimal value for the confidence threshold changes with minimum support (from 0.001 for a support of 0.0001 to 0.1 at 0.00003).

Figure 12 shows the mean absolute error of the recommendations by the coverage produced by the algorithms for different parameters. Again, for the support-confidence framework performance deteriorates significantly for very small misspecifications of the confidence threshold. The repeat-buying algorithm shows a more robust behavior with respect to changes to its one threshold which represents the maximum expected type II error rate β in the produced recommendation lists. In contrast to minimum support and minimum confidence, β is independent of the structure and the properties of the analyzed data set. β is a very convenient parameter since it can be interpreted as the proportion of incorrect recommendations in a recommendation list. Figure 13 shows the dependence of the $average\ observed\ \beta$ on the threshold of the repeat-buying algorithm. With the exception of the threshold values 0, 0.1, and 0.2 (where only few lists are produced) the $average\ observed\ \beta$ is, as expected by the model, below the threshold (the $maximum\ expected\ \beta$). This property gives the algorithm a very stable behavior.

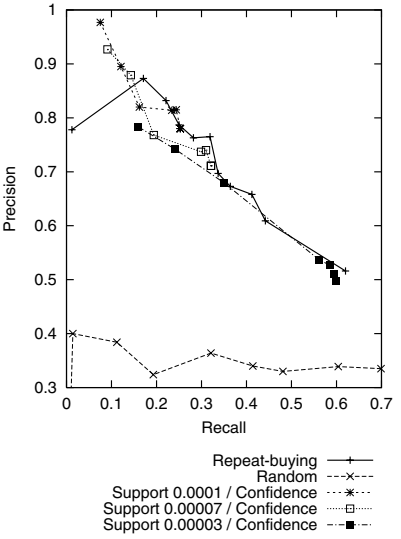


Fig. 10. Precision by recall plot

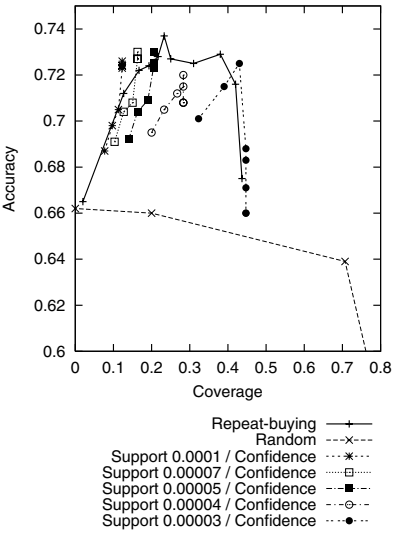


Fig. 11. Accuracy by coverage

8 Conclusion

Evaluation of the performance of recommender algorithms is an important part of the knowledge discovery process. However, for the data mining part of recommender systems the question of how well found patterns match the user's concept of useful recommendations is often neglected. In this paper we studied this question by comparing recommendations by human peers with the recommendations produced by two recommender algorithms. In the following we summarize the results and remaining research questions:

- The result of the presented evaluation supports the widely accepted assumption that frequent itemsets from purchase histories or from Web usage data represent useful recommendations and therefore support the social process of recommending items. With a well-parameterized algorithm an accuracy of more than 70% and a precision between 60% and 90% have been reached in this study.
- Association rules are free of model assumptions, whereas the repeat-buying algorithm requires several quite strong model assumptions on user behavior which are hard to verify and which invite critic on the validity of the model. However, the assumptions tend to hold approximately for a wide range of applications for consumer products [11]. The good results of the presented recommender algorithm based on repeat-buying is strong evidence that usage patterns in the online world can be described with similar models as consumption behavior in the physical world (e.g. stores).
- The results of both algorithms, when properly tuned, are quite similar. However, the repeat-buying algorithm uses only one parameter which is independent of the data

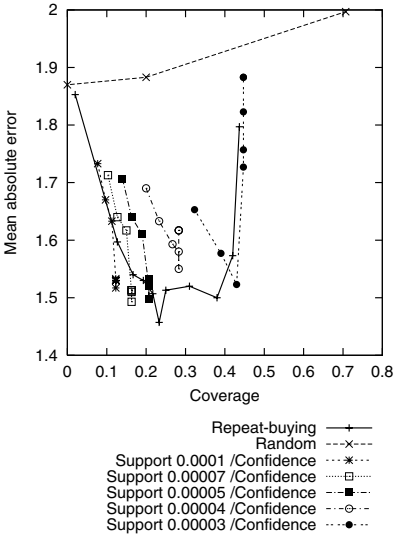


Fig. 12. Mean absolute error by coverage

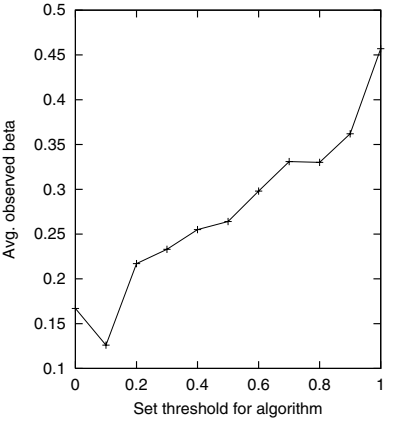


Fig. 13. β by threshold of the repeat-buying algorithm

set and has a simple interpretation, whereas the association rule algorithm uses two parameters which strongly depend on each other and on properties of the data set. Furthermore, the repeat-buying algorithm seems to be more robust with regard to a misspecification of its parameter, whereas the association rule algorithm is more flexible and allows fine tuning.

– Both algorithms studied in this paper have a computational complexity of quadratic order. However, for incremental updates the repeat-buying algorithm has a computational complexity of quadratic order in the number of updated items, whereas the simple association rule algorithm is of quadratic complexity on all items. For applications with millions of items the repeat-buying algorithm's possibility to perform efficient incremental updates is essential. Strategies for incremental updates of association rules need to be developed.
– The parameters for the association rule algorithm depend on the size and the structure of the data set, whereas the single parameter of the repeat-buying algorithm, the sample mean, is robust and sufficient. It is independent of the data set, because in a statistical sense the LSD-model is an approximation to a zero truncated negative binomial distribution. For the automatic operation in a dynamically changing environment this seems to be a considerable advantage.

Since we only used one data set for this paper, additional studies of other data sets from different sources are needed to confirm our findings. We are currently working on the evaluation of large data sets from a B2B-merchant for computer accessories and from the South-West German Library Network.

References

1. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. In: From Data Mining to Knowledge Discovery: An Overview. MIT Press, Cambridge, MA (1996) 1–36
2. Resnick, P., Varian, H.R.: Recommender systems. Communications of the ACM **40** (1997) 56–58
3. Shardanand, U., Maes, P.: Social information filtering: Algorithms for automating 'word of mouth'. In: Conference proceedings on Human factors in computing systems (CHI'95), Denver, CO, ACM Press/Addison-Wesley Publishing Co. (1995) 210–217
4. Spiliopoulou, M.: Web usage mining for web site evaluation. Communications of the ACM **43** (2000) 127–134
5. Mobasher, B., Cooley, R., Srivastava, J.: Automatic personalization based on web usage mining. Communications of the ACM **43** (2000) 142–151
6. Wirth, R., Hipp, J.: CRISP-DM: Towards a standard process modell for data mining. In: Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining, Manchester, UK (2000)
7. Lawrence, R.D., Almasi, G.S., Kotlyar, V., Viveros, M.S., Duri, S.: Personalization of supermarket product recommendations. Data Mining and Knowledge Discovery **5** (2001) 11–32
8. Quadt, A.: Personalisierung im e-commerce. Diplomarbeit, AIFB, Universität Karlsruhe (TH), D-76128 Karlsruhe, Germany (2001)
9. Adamo, J.M.: Data Mining for Association Rules and Sequential Patterns. Springer, New York (2001)
10. Mild, A., Natter, M.: Collaborative filtering or regression models for internet recommendation systems? Journal of Targeting, Measurement and Analysis for Marketing **10** (2002) 304–313
11. Ehrenberg, A.S.C.: Repeat-Buying: Facts, Theory and Application. Charles Griffin & Company Ltd., London (1988)
12. Tan, P.N., Kumar, V.: Discovery of web robot sessions based on their navigational patterns. Data Mining and Knowledge Discovery **6** (2002) 9–35
13. Cooley, R., Mobasher, B., Srivastava, J.: Data preparation for mining world wide web browsing patterns. Journal of Knowledge and Information Systems **1** (1999) 5–32
14. Cooley, R.W.: Web usage mining: Discovery and application of interesting patterns from web data. Ph. d. thesis, Graduate School of the University of Minnesota, University of Minnesota (2000)
15. Berendt, B., Mobasher, B., Spiliopoulou, M., Nakagawa, M.: The impact of site structure and user environment on session reconstruction in web usage analysis. In: Proceedings of the 4th WebKDD 2002 Workshop, at the ACM-SIGKDD Conference on Knowledge Discovery in Databases (KDD'2002), Edmonton, Alberta, Canada (2002)
16. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: Explaining collaborative filtering recommendations. In: Proceedings of the ACM 2000 Conference on Computer Supported Cooperative Work. (2000) 241–250
17. Kohavi, R., Provost, F.: Glossary of terms. Machine Learning **30** (1988) 271–274
18. Salton, G., McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill, New York (1983)
19. van Rijsbergen, C.: Information retrieval. Butterworth, London (1979)
20. Mobasher, B., Dai, H., Tao Luo, M.N.: Discovery and evaluation of aggregate usage profiles for web personalization. Data Mining and Knowledge Discovery **6** (2002) 61–82
21. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: Proceedings of the 1999 Conference on Research and Development in Information Retrieval. (1999) 230–237

22. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Application of dimensionality reduction in recommender systems–a case study. In: ACM WebKDD 2000 Web Mining for E-Commerce Workshop. (2000)
23. Mobasher, B., Dai, H., Luo, T., Nakagawa, M., Sun, Y., Wiltshire, J.: Discovery of aggregate usage profiles for web personalization. In: ACM WebKDD 2000 Web Mining for E-Commerce Workshop. (2000)
24. Vucetic, S., Obradovic, Z.: A regression-based approach for scaling-up personalized recommender systems in e-commerce. In: ACM WebKDD 2000 Web Mining for E-Commerce Workshop. (2000)
25. Yu, K., Xu, X., Ester, M., Kriegel, H.P.: Selecting relevant instances for efficient accurate collaborative filtering. In: Proceedings of the 10th CIKM, ACM Press (2001) 239–246
26. Lin, W., Alvarez, S.A., Ruiz, C.: Efficient adaptive-support association rule mining for recommender systems. Data Mining and Knowledge Discovery **6** (2002) 83–105
27. Agrawal, R., Imielinski, T., Swami, A.: Mining associations between sets of items in large databases. In: Proc. of the ACM SIGMOD Int'l Conference on Management of Data, Washington D.C. (1993) 207–216
28. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: Proc. 20th Int. Conf. Very Large Data Bases, VLDB, Santiago, Chile (1994) 487–499
29. Brin, S., Motwani, R., Ullman, J.D., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. In: SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, USA (1997) 255–264
30. Aggarwal, C.C., Yu, P.S.: A new framework for itemset generation. In: PODS 98, Symposium on Principles of Database Systems, Seattle, WA, USA (1998) 18–24
31. Bayardo, R., Agrawal, R., Gunopulos, D.: Constraint-based rule mining in large, dense databases. Data Mining and Knowledge Discovery **4** (2000) 217–240
32. Geyer-Schulz, A., Hahsler, M.: A customer purchase incidence model applied to recommender systems. In Kohavi, R., Masand, B.M., Spiliopoulou, M., Srivastava, J., eds.: WEBKDD 2001 Mining Web Log Data Accross All Customer Touch Points. Volume 2356 of LNAI., Berlin, Springer (2002) 25–47
33. Cheung, D.W., Jiawei, H., Ng, V.T., Wong, C.Y.: Maintenance of discovered association rules in large databases: An incremental updating technique. In: Proceedings of the 12th International Conference on Data Engineering, 1996. New Orleans., Piscataway, IEEE (1996) 106–114
34. Ng, K.K., Lam, W.: Updating of association rules dynamically. In: International Symposium on Database Applications in Non-Traditional Environments, 1999 (DANTE'99) Proceedings., Piscataway, IEEE (2000) 84–91
35. Zheng, Z., Kohavi, R., Mason, L.: Real world performance of association rule algorithms. In Provost, F., Srikant, R., eds.: Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining (ACM-SIGKDD), ACM Press (2001) 401–406

The Impact of Site Structure and User Environment on Session Reconstruction in Web Usage Analysis

Bettina Berendt[1], Bamshad Mobasher[2], Miki Nakagawa[2], and Myra Spiliopoulou[3⋆]

[1] Humboldt University Berlin,
Inst. of Information Systems,
berendt@wiwi.hu-berlin.de
[2] DePaul University Chicago,
Dept. of Computer Science,
{mobasher|mnakagawa}@cs.depaul.edu
[3] Otto von Guericke University Magdeburg,
Faculty of Computer Science,
myra@iti.cs.uni-magdeburg.de

Abstract. The analysis of user behavior on the Web presupposes a reliable reconstruction of the users' navigational activities. Cookies and server-generated session identifiers have been designed to allow an accurate session reconstruction. However, in the absence of reliable methods, analysts must employ heuristics (a) to identify unique visitors to a site, and (b) to distinguish among the activities of such users during independent sessions. The characteristics of the site, such as the site structure, as well as the methods used for data collection (e.g., the existence of cookies and reliable synchronization across multiple servers) may necessitate the use of different types of heuristics. In this study, we extend our work on the reliability of sessionizing mechanisms, by investigating the impact of site structure on the quality of constructed sessions. Specifically, we juxtapose sessionizing on a frame-based and a frame-free version of a site. We investigate the behavior of cookies, server-generated session identification, and heuristics that exploit session duration, page stay time and page linkage. Different measures of session reconstruction quality, as well as experiments on the impact on the prediction of frequent entry and exit pages, show that different reconstruction heuristics can be recommended depending on the characteristics of the site. We also present first results on the impact of session reconstruction heuristics on predictive applications such as Web personalization.

Keywords: Data preparation, sessionization heuristics, Web usage mining

⋆ This research was done while at Leipzig Graduate School of Management, Dept. of E-Business.

O.R. Zaïane et al. (Eds.): WEBKDD 2002, LNAI 2703, pp. 159–179, 2003.

1 Introduction

The quality of the patterns discovered in data analysis depends on the quality of the data on which mining is performed. In Web usage analysis, these data are the sessions of the site visitors. The reliable reconstruction of the visitors' activities in a Web site involves a correct mapping of activities to different individuals and a correct separation of the activities belonging to different visits of the same individual.

In [BMSW01,SMBN03], we describe how the two aspects of reliable session reconstruction are supported by (a) proactive mechanisms that enforce correct mappings during the activities of each visitor and (b) reactive heuristics that perform the mappings a posteriori. Cookie identifiers and session identifiers generated by Web application servers belong to the first category. The specification of upper thresholds on total visit time or on total page stay time are examples of heuristics of the second category. A collection and discussion of such heuristics can be found in [CMS99]. Section 2 of the current paper contains a formalization of these heuristics.

Padmanabhan, Zheng, and Kimbrough stress the importance of correct session reconstruction by comparing different definitions of the notion of "session" and comparing their predictive accuracy [PZK01,ZPK03]. However, they do not elaborate on the performance of different mechanisms for session reconstruction.

In [BMSW01], we have proposed a set of measures for the comparison of sessions generated by different heuristics (see section 3). In [BMSW01,SMBN03], we used these measures to compare the results of proactive and reactive heuristics on a frame-based site. Our experiments showed that proactive mechanisms (like cookies) allow for a much more reliable session reconstruction than reactive ones.

Increasingly, commercial and (to a lesser degree) non-commercial sites have been relying on cookies as a mechanism for the identification of unique users. However, there is still a large proportion of sites that do not use such proactive mechanisms. This is in part due to regulations and self-regulations on e-privacy, particularly in Europe, and in part due to a lack of adequate infrastructural support in data collection and analysis. Furthermore, cookies, by themselves, are not adequate for the correct splitting of a visitor's activities into sessions, and even fewer sites currently use server-supported proactive mechanisms for sessionization. As long as no proactive, privacy-conforming mechanism has become commonplace among Web servers, reactive heuristics will therefore remain essential for reliable session reconstruction for usage analysis.

In our previous work [SMBN03], we have observed that the framesets of a frame-based site have a serious impact on the mapping of the activities of each user to distinct sessions. Hence, in this study, we investigate the performance of the sessionizing heuristics for a frame-based and a frame-free version of a site (section 4). In section 5, we elaborate on the influence that the user environment has on the heuristics' performance, in particular when the relation between visitor and IP address is not 1:1.

Since session reconstruction is a preparatory step for data analysis, the "performance" of the heuristics should refer to the predictive power of the sessions

they generate. Section 6 describes experiments on the effects on the data mining tasks of (a) predicting entry and exit pages and (b) recommending a page based on usage profiles. The last section concludes our study.

2 Heuristic Methods for Session Reconstruction

Heuristic methods for session reconstruction must fulfill two tasks: First, all activities performed by the same physical person should be grouped together. Second, all activities belonging to the same visit should be placed into the same group. Knowledge about a user's identity is not necessary to fulfill these tasks. However, a mechanism for distinguishing among different users is indeed needed.

In accordance with W3C (1999), we term as *(server) session* or *visit* the group of activities performed by a user from the moment she enters the site to the moment she leaves it. Since a user may visit a site more than once, the Web server log records multiple sessions for each user. We use the name *user activity log* for the sequence of logged activities belonging to the same user. Thus, *sessionizing* is the process of segmenting the user activity log of each user into sessions. A *sessionization heuristic* is a method for performing such a segmentation on the basis of assumption about users' behavior or the site characteristics.

The goal of a heuristic is the faithful reconstruction of the *real sessions*, where a real session is the sequence of activities performed by one user during one visit at the site. We denote the dataset of real sessions as $\mathcal{R}$. A sessionization heuristic h attempts to assign activities to users and to identify the ends of each user visit, i.e. to partition sequences of activities of the same user into sessions. The result is a dataset of *constructed sessions*, which we denote as $\mathcal{C} \equiv \mathcal{C}_h$. For the ideal heuristic, $\mathcal{C} \equiv \mathcal{C}_h = \mathcal{R}$.

2.1 Mapping Activities to Users and Segmenting into Sessions

The analysis of Web usage does not require knowledge about a user's identity. However, it is necessary to distinguish among different users. The information available according to the HTTP standard is not adequate to distinguish among users from the same host, proxy, or anonymizer. The most widespread remedy amounts to the usage of cookies. A persistent cookie is a unique identifier assigned by the Web server to each client agent accessing the site (or other sites associated with the same domain) for the first time. This identifier is stored on the client side and transmitted back to the server upon subsequent visits to the server by the same client. A cookie is a *proactive data preparation strategy* because the assignment of a user identification to requests is taken care of *while* the user accesses the site. However, while a cookie provides for user identification across multiple visits to the site, it does not mark these visits' boundaries.

The proactive approach to session identification may also involve the use of *embedded session IDs*. Such session IDs are implemented as an extension of the

Web server which assigns a unique identifier to each active client process accessing the server. This identifier is attached to each request made by the user's client to the server (e.g., by URL rewriting), thus allowing for the unique assignment of requests to users during one visit. The identifier expires when the user's client process is terminated, when the connection is broken, or when a timeout occurs. Its expiration determines the end of the session. Other proactive strategies include user authentication / registration and client agent data collection.

In contrast to proactive strategies, *reactive strategies* reconstruct the assignment of a user (or session) identification to requests *after* the requests have been recorded, based on the "user environment" information recorded in the Web server's log.

The most widespread log formats for HTTP servers are the W3C common and extended log file formats. In the common log file format[1], the only recorded data related to a user as a person are the IP address or DNS hostname of the user's host or proxy. The extended log file format[2] also allows the recording of the user's software agent (browser or batch client) that performs the requests on her behalf, as well as the "referrer" URL, i.e. the page from which a request was initiated. However, partitioning by IP+agent is not guaranteed to differentiate between the activities of different users correctly, since several users may be accessing the server from the same IP+agent. Also, it cannot recognize session boundaries.

Therefore, a second step is required that generates a further partitioning into (constructed) sessions. Since this second step is needed for logs partitioned by cookies as well as for logs partitioned by IP+agent, the same *sessionization heuristics* can be employed. These too are reactive strategies.

2.2 A Selection of Sessionization Heuristics

In the present study, we evaluate the performance of the following heuristics:

h1: Time-oriented heuristic: The duration of a session may not exceed a threshold θ.
This heuristic has its origins in research on the mean inactivity time within a site [CP95].

h2: Time-oriented heuristic: The time spent on a page may not exceed a threshold δ.
Heuristics of this type are used in [CMS99,SF99].

href: Referrer-based heuristic: Let p and q be two consecutive page requests, with p belonging to a session S. Let t_p and t_q denote the timestamps for p and q, respectively. Then, q will be added to S if the referrer for q was previously invoked within S, or if the referrer is undefined and $(t_q - t_p) \leq \Delta$, for a specified time delay Δ. Otherwise, q is added to a new constructed session.

[1] see http://www.w3.org/Daemon/User/Config/Logging.html
[2] see http://iishelp.web.cern.ch/IISHelp/iis/htm/core/iiintlg.htm

The "undefined" referrer ("–" in the log) is usually caused by a server-dependent process. In many logs, an undefined referrer may be recorded in various situations: (1) As the referrer of the start page, or of a page that was entered after a brief excursion to a sub-site or a foreign server. This may happen, for example, because a site does not record external referrers. (2) As the referrer of a typed-in or bookmarked URL. (3) When a frameset page is reloaded in mid-session. (4) For all these pages, when they are reached via the back button during the real session. (5) In a frame-based site: as the referrer of the first frames that are loaded when the start page containing the top frameset is requested. (6) When access to a page was invoked by certain external processes such as a hyperlink from an email client or from within a non-HTML document.
The time delay Δ in the above definition is necessary to allow for proper loading of frameset pages whose referrer is undefined, and to account for other situations resulting in mid-session requests with undefined referrers.

In the present study, we consider the three heuristics in the two settings *frame-based* and *frame-free*, by comparing the results of the previous setting with cookie identifiers in a site with frames with cookie-identified data from the same site in a frame-free version. In both settings, we have used the standard values $\theta = 30$ minutes maximum total duration for **h1**, and $\delta = 10$ minutes maximum page stay time for **h2** (see [CP95,CMS99,SF99]). Previous experiments indicate a high robustness of these heuristics with respect to variations in the threshold parameters, and superior performance for the two values chosen. In both settings, we have used a default value of 10 seconds for **href**'s Δ, and also investigated the effect of varying this value.

2.3 Factors That Affect the Session (Re)construction Process

After the introduction of the heuristics, we can take a step back to place our current study into a more general framework. As we have seen in section 2.1, session reconstruction involves two processes: the mapping of activities to different users ("user mapping"), and the segmentation into single visits or sessions ("session segmentation"). Various factors influence the quality of these two processes. One of these is the method used for reconstruction, the *heuristic* chosen. However, characteristics of the user population in the log as well as of the site itself are also likely to affect the quality of reconstruction. In a series of studies, we have operationalized and investigated these factors. We have concentrated on two central aspects of user population and site characteristics: the *user environment*, and the *site structure*.

In [BMSW01], we investigated the influence of the factor *heuristic*, using the values **h1-30**, **h2-10**, and an earlier version of **href**, as described in section 2.2. In [BMSW01], we concentrated on the influence on session segmentation: Using data from a frame-based site with cookie identifiers, we investigated the performance of the three heuristics and discussed possible reasons for their differential performance. This allowed us to establish a baseline for the influence of the *heuristic* on session segmentation.

In [SMBN03], we used the same heuristics and compared their performance on data with prior cookie-based user identification, to their performance on data with prior IP+agent partitioning. We thus distinguished the values "cookie" and "u-ipa" of the factor *user environment*, and investigated how this affected session reconstruction. We were thus able to analyze the loss in reconstruction quality resulting from adding the task of user mapping to the task of session segmentation.

In the present paper, we consider *site structure* as an additional factor, and we vary both this factor and the *heuristic*. We use data from a frame-based and from a frame-free version of the same site, both with cookie identifiers, and investigate the performance of the three heuristics. (The previous studies both operated on data from a frame-based site.) Concentrating on the "cookie" setting, we are thus able to analyze the influence of *site structure* on session segmentation[3], as well as possible interactions with *heuristic*. This analysis is contained in section 4 of the present paper. It is complemented by analyses of the impact of heuristic and site structure on mining applications in section 6.

In addition, the paper extends the analysis, in [SMBN03], of the *user environment*. In that paper, we have investigated only two values, "cookie" and "u-ipa". In a next step, "u-ipa" can itself be further differentiated. In section 5, we discuss the constellations that may occur in a log and the types of errors they may cause. We show that the logs we have used so far present very favorable conditions with regard to these sources of errors. We give an outlook on experimental settings suited to analyzing these errors in more detail.

In order to measure the impact of these factors on session reconstruction quality, we first need to define measures of reconstruction quality.

3 Measures of Session Reconstruction Quality

Intuitively, the perfect heuristic would reconstruct all sessions by placing all activities of each user during each visit—and only these—into the same session. Reactive heuristics do not have adequate information for such a perfect assignment of activities to sessions. Thus, it is necessary to quantify the performance of each heuristic with respect to the quality of *all* sessions it builds.

The measures we use quantify the successful mappings of real sessions to constructed sessions, i.e. the "reconstructions of real sessions". A measure M evaluates a heuristic h based on the difference between $\mathcal{C}_h$ and $\mathcal{R}$. It assigns to h a value $M(h) \in [0, 1]$ such that the score for the perfect heuristic ph is $M(ph) = 1$.

In [BMSW01,SMBN03], we have proposed four "categorical" measures that reflect the number of real sessions that are reconstructed by a heuristic *in their entirety*, and two "gradual" measures that take account of the extent to which the real sessions are reconstructed. Here, we reinterpret the categorical measures as recall measures, and extend the framework by the corresponding precision measures. This allows a more differentiated analysis of reconstruction quality.

[3] In general, the site structure can also influence the process of user mapping.

Categorical Measures. Categorical measures enumerate the real sessions that were recognized by the heuristics as distinct visits and were thus mapped *into* constructed sessions, i.e. they are contained in some constructed session.

- The *complete reconstruction* measure $M_{cr}(h)$ returns the number of real sessions contained in some constructed session, divided by the total number of real sessions. A session r is contained in a session c if and only if all its elements are in c, in their correct order, with no intervening foreign elements.

This measure is not very specific, since any number of real sessions may be contained in one constructed session without affecting its value. Therefore, further elaboration is necessary. We consider two types of refinement: On the one hand, we design more restrictive measures, in which the entry or exit page of the real session has been identified as such in the constructed session. On the other hand, we juxtapose the number of real sessions thus considered against either all real sessions or all constructed sessions.

The first type of refinement reflects the fact that the correct identification of the entry or the exit page is essential for many applications. If both the entry and the exit page of a real session are guessed correctly, then the corresponding constructed session is identical to the real one.

The second type of refinement corresponds to the notions of recall and precision. Our algorithms ensure that if a reconstructed session contains a real session and has the same entry (exit) page, then there is only one such constructed session. Therefore, this can be interpreted to mean that the constructed session is the (unique) "correct guess" of that real session. So the corresponding measures can be interpreted as *recall* measures: the number of correct guesses divided by the total number of correct items $|\mathcal{R}|$. We complement this by the corresponding *precision* measures: the number of correct guesses divided by the total number of guesses $|\mathcal{C}_h|$.

We therefore define:

- *complete reconstruction with correct entry page – recall*:
  $M^{recall}_{cr,entry}(h)$ is the number of real sessions mapped into a constructed session with the same entry page as the real session, divided by the number of real sessions.

- *complete reconstruction with correct exit page – recall*:
  $M^{recall}_{cr,exit}(h)$ is the number of real sessions mapped into a constructed session with the same exit page as the real session, divided by the number of real sessions.

- *identical reconstruction – recall*:
  $M^{recall}_{cr,entry-exit}(h)$ is the number of real sessions that appear in $\mathcal{C}_h$, i.e. the intersection of $\mathcal{R}$ and $\mathcal{C}_h$, divided by the number of real sessions.

- *complete reconstruction with correct entry page – precision*:
  $M^{precision}_{cr,entry}(h)$ is the number of constructed sessions that contain a real session

and have the same entry page with it, divided by the number of constructed sessions.

– *complete reconstruction with correct exit page – precision*:
$M_{cr,exit}^{precision}(h)$ is the number of constructed sessions that contain a real session and have the same exit page with it, divided by the number of constructed sessions.

– *identical reconstruction – precision*:
$M_{cr,entry-exit}^{precision}(h)$ is the number of sessions in the intersection of $\mathcal{R}$ and $\mathcal{C}_h$, divided by the number of constructed sessions.

Categorical measures are relevant for applications in which the faithful reconstruction of the real sessions in their entirety is necessary for the analysis. This includes the evaluation of user satisfaction with a site, and the analysis of the improvement potential of the site as a whole.

Gradual Measures. For some application domains, categorical measures are too restrictive. For example, consider the page prefetching problem, in which the next request must be predicted given the requests performed thus far. This only requires the correct reconstruction of previous requests in the same session, usually without recourse to their exact sequence. Similarly, for market basket analysis or recommender systems, the identification of as many co-occurrences of pages as possible is more relevant than the correct identification of the exact sequence of actions, or the correct identification of entry and exit pages. For such applications, we use *gradual measures*.

Gradual measures are based on the overlap between real and reconstructed sessions. For each real session, we find the constructed session that has the maximum number of common elements with it. We normalize this value by dividing it by the length of the real session. Then, we define the *average maximal degree of overlap* $M_o(h)$ as the average of these overlap values over all real sessions.

The measure $M_o(h)$ does not take the number of dissimilar elements between real and constructed session into account. Hence, a heuristic that produces only one constructed session will acquire the highest score 1 for $M_o(h)$. To alleviate this problem, we consider the similarity between real and constructed sessions: For each real session, we again find the constructed session that has the maximum number of common elements with it, but we divide it by the number of elements in the union of real and constructed session. The *average maximal degree of similarity* $M_s(h)$ is the average of these values over all real sessions.

4 Impact of the Site Structure on Session Reconstruction Quality

We first compared the set of reconstructed sessions produced by each heuristic to the "real" sessions. In the analyzed site, a cookie-based mechanism was used

for user identification, and session IDs for splitting a user's activities into sessions. This combination defined the reference set $\mathcal{R}$ of real sessions. The session identifiers were then removed, and each of the heuristics h described in section 2.2 was employed to produce a set of constructed sessions $\mathcal{C}_h$. By juxtaposing the reconstruction results for a frame-based and a frame-free version of a site, we show that the frame-free version allows for more reliable session reconstruction. Different heuristics are affected by framesets to a different extent.

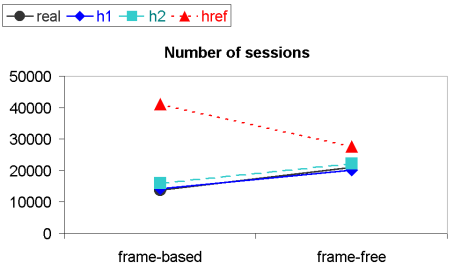


Fig. 1. Base statistics for the frame-based and the frame-free sites (1).

The two datasets describe the usage of the same university site, once in a frame-based, and once in a frame-free version. They contained 174660 and 115434 requests, respectively.

The preprocessing of the data included removal of accesses from known robots and well-behaved robots (those accessing the `robots.txt` page). We expect that the remaining robots constitute a negligible percentage of sessions. Between 1 and 2% of all users rejected cookies; their requests were removed.

4.1 Session Statistics for a Frame-Based and a Frame-Free Site

Figures 1 and 2 show the basic statistics for the two site versions.

In Fig. 2, we report median values because earlier experiments [BMSW01, SMBN03] showed that averages were not representative of the distribution of values in the frame-based dataset. The discrepancy between median and average values holds both for the frame-based and the frame-free version and is most remarkable for session duration. It is apparent that a heuristic making a good approximation of the median values of the real sessions captures the characteristics of the real sessions much better than one that only approximates the average values.

With respect to the approximation of average values, the three heuristics behave in similar ways for the frame-based and the frame-free version of the site. The value distributions they produce are smoother than the distribution of the real values in the sense that averages and medians lie closer together for constructed than for real sessions. In particular: (a) All heuristics underestimate

the average session duration in both the frame-based and the frame-free version. (b) All heuristics underestimate the average page stay time for both site versions. (c) **h1** and **h2** return better approximations of average session length than **href**, with **h1** returning the best approximations.

With respect to the approximation of median values, the three heuristics behave in similar ways for the frame-based and the frame-free version of the site. In particular: (a) Heuristic **h2** makes the best approximation of all median values in both the frame-based and the frame-free version. (b) Heuristic **h1** overestimates all median values, while **href** underestimates them.

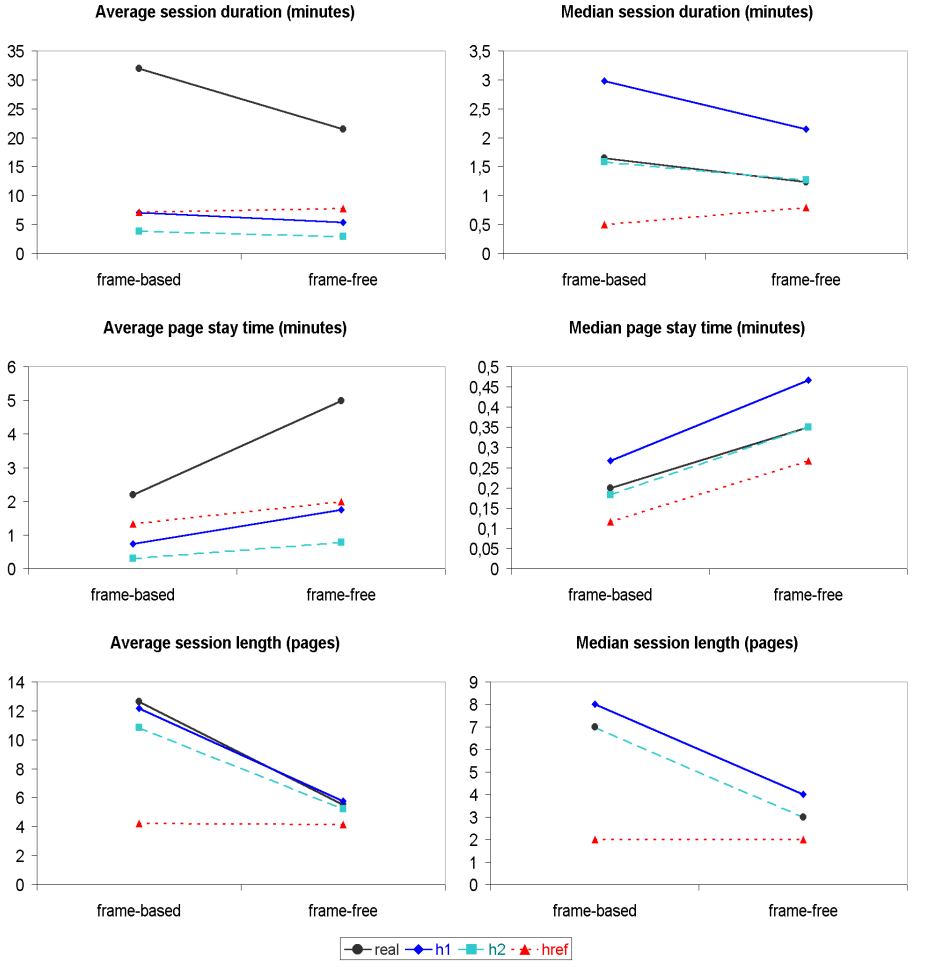


Fig. 2. Base statistics for the frame-based and the frame-free sites (2).

The heuristic **href** makes better approximations of the medians in the frame-free than in the frame-based version of the site. This was expected; pages in the frame-free site have fewer objects than those in the frame-based site, so that **href** encounters fewer misleading referrers.

Heuristic **h1** provides the best approximation of the number of real sessions. Heuristic **h2** overestimates this number for both the frame-based and the frame-free version. Overestimation is a more severe problem for **href**: It produces more than three times as many sessions as there actually are. Its performance is much better in the frame-free version, where "only" 25% more sessions are produced.

To explain the performance of **href**, we cross-compare the disproportionately large number of sessions, the very low median values for session duration, page stay and session length in pages, and the low average session length for this heuristic. Those numbers indicate that **href** produces a very large number of tiny reconstructed sessions by splitting real sessions. In the frame-based version of the site, the median length of real sessions is 7, and the corresponding median for the sessions produced by **href** is 2, implying that each session in $\mathcal{R}^{fb}$ corresponds to around 3 sessions in $\mathcal{C}^{fb}_{href}$. In the frame-free version, the median length of real sessions is only 3, so that the tiny sessions produced by **href** produce better approximations for the statistics upon the real sessions.

In terms of applicability of the heuristics, the basic statistics indicate that the referrer heuristic should not be used in a frame-based site. The two time-based heuristics are less affected by the presence of framesets and can be used in both types of site.

4.2 Comparing Session Contents

The base statistics reflect the properties of the datasets of reconstructed sessions. However, they do not indicate to what extent the reconstructed sessions are the same as the real ones. The measures described in section 3 analyze session content in this way.

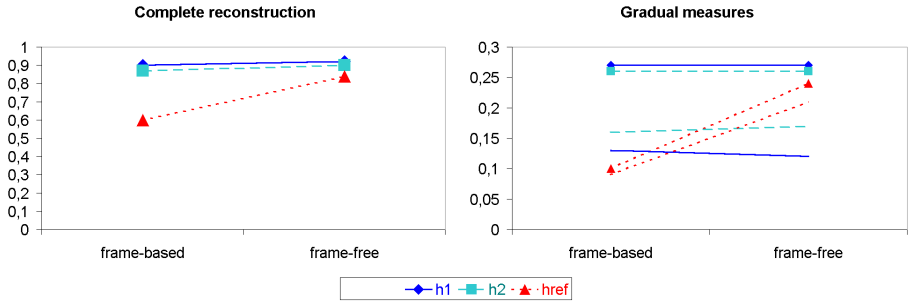


Fig. 3. Complete reconstruction and gradual measures. The gradual measures are overlap M_o (lines with end markers) and similarity M_s (lines without end markers).

Figures 3 and 4 show the values of these measures for the two datasets. Results for $M_{cr,exit}$ are virtually identical to those for $M_{cr,entry}$, and are therefore omitted.

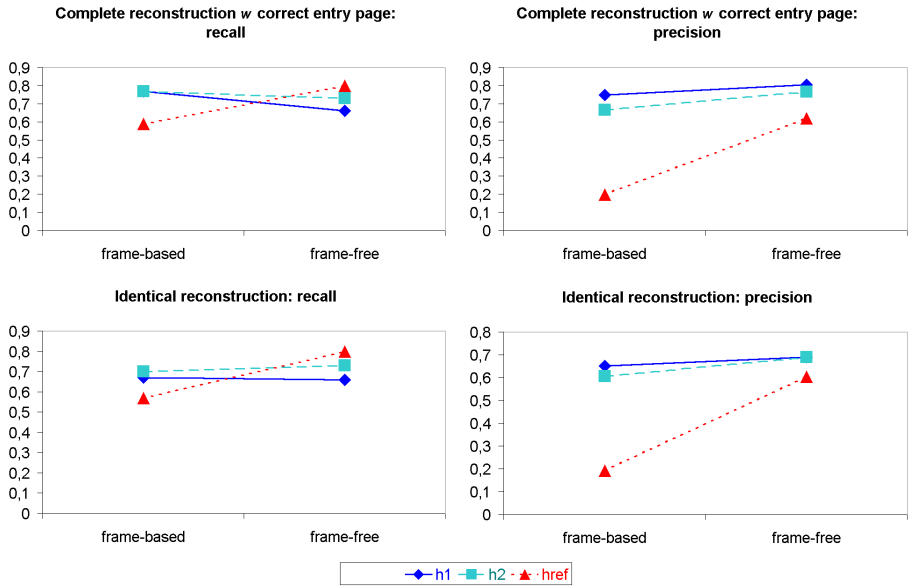


Fig. 4. Recall and precision for measures $M_{cr,entry}$ (top) and $M_{cr,entry-exit}$ (bottom).

The quality of the reconstructed sessions with respect to our measures differs between heuristics. Moreover, the presence of framesets does affect the results. In particular: (a) Heuristic **h1** has very similar scores for the frame-based and the frame-free version, and is thus "frame-insensitive". (b) Heuristic **h2** has higher scores for the frame-free version. (c) For the frame-based version, heuristic **h2** has best scores for the most restrictive measures ($M^{recall}_{cr,entry}$, $M^{recall}_{cr,entry-exit}$, and M_s), while **h1** performs better for the less restrictive measures M_{cr} and M_o. However, the precision of **h2** is lower than that of **h1**. (d) Heuristic **href** has much higher scores for the frame-free version than for the frame-based one. It outperforms the other two heuristics in the scores of the three categorical recall measures and on the gradual measure M_s. Its precision increases greatly relative to the frame-based version. Note that the similarity measure M_s shows the same behavior across site structures and heuristics as the measure $M^{recall}_{cr,entry-exit}$. So in the frame-free site, **href** correctly identifies the highest proportion of real sessions in their entirety, and it retains the highest proportion of real session fragments.

In summary, heuristics **h1** and **h2** are most appropriate for sites providing both a frame-based and a frame-free version of their content. The good perfor-

mance of **href** on the frame-free version suggests that it is appropriate if the sessions in the frame-free version are small.

4.3 The Impact of Session Length

Figure 2 has shown that most of the real sessions are shorter than the average. Hence, it is of interest how effective each heuristic is in reconstructing the short sessions of the Web server log. Figure 5 gives an overview of the distribution of session lengths in the frame-free site.

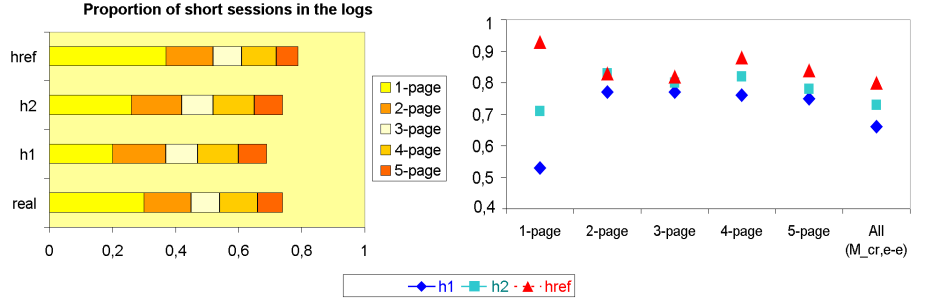


Fig. 5. Short sessions (left) and identically reconstructed sessions (right) in the frame-free site version. M_cr,e-e $= M^{recall}_{cr,entry-exit}$.

Here, **href** produces a large number of one-page sessions, but the percentage of sessions produced for each length is similar to the percentage of sessions of this length in the log of real sessions.

Similarly to the base statistics, the statistics per session length do not guarantee that the real and the reconstructed sessions of length n are identical. We have computed the complete matches between real and constructed session, i.e. the percentage of real sessions that appear in the log of reconstructed sessions. The results for sessions with one to five pages are shown on the right hand side of Fig. 5. The three proportions associated with the heuristics correspond to the value of the $M^{recall}_{cr,entry-exit}$ measure, as defined in section 3, but they are taken over the sessions of a given length instead of over the whole dataset of real sessions. This value, i.e. $M^{recall}_{cr,entry-exit}$ taken over the whole dataset of real sessions, is repeated in the rightmost column of the figure. Fig. 5 shows that the performance improvement of the referrer heuristic is due to the complete reconstruction of most small sessions.

Figure 6 shows the measurements for the frame-based version. The figure show that the percentage of small sessions for the time-based heuristics is close to the percentage of real sessions of the same size. Here, **href** produces a disproportionately large number of one-page sessions. This means that the relationships shown in Fig. 5 are even more strongly pronounced in the frame-based site.

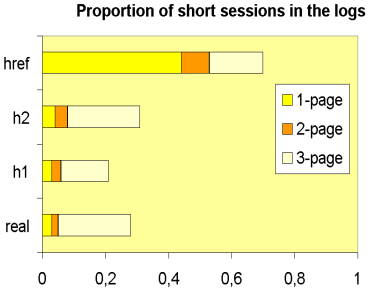


Fig. 6. Short sessions in the frame-based site.

These results indicate that the parameter setting for this heuristic favors small session lengths. To investigate this further, we varied (in the frame-free version) **href**'s parameter Δ, i.e., the allowable time gap between consecutive log entries with undefined referrer within one session. Increasing Δ led to a decrease in the ratio of the number of constructed sessions to the number of real sessions. With Δ approaching 10 minutes, this ratio approached 2 for the frame-based site, and 1 for the frame-free site. Also, the impact of varying Δ in the range 0–30 seconds was much larger for the frame-based site. This is not surprising given that if a top frame page has an undefined referrer, then so do all the components.

With increasing Δ, the median duration of the constructed session converges towards the median of the real sessions for large Δs; the median of the constructed session sizes remains constant while the average increases towards the average of real session sizes. This implies that larger Δs allow for the reconstruction of the few long sessions, too.

The top value of Δ that we investigated, 10 minutes, is equal to **h2**'s parameter δ, the maximum page stay time on *any* page. With Δ equal to δ, the two heuristics treat pages with undefined referrers alike, but pages with defined referrers differently. **h2** splits at a page with a defined referrer if the elapsed time was too long. This bears the risk of misinterpreting a page that really was inspected for a long time. **href** splits if the referrer was not in the current session. This bears the risk of misinterpretation if, for example, a request was delivered from the cache. These two effects have to be investigated in their interaction in more detail, and a combined heuristic may be desirable.

5 Impact of the User Environment on Session Reconstruction Quality

In this section, we focus on the influence of the user environment information concerning IP and agent. In an "ideal" setting, each user would have her own individual computer with a fixed IP address, and use the same browser for a longer time. This would make IP+agent equivalent to a cookie. However, in many real settings, the relation is not one-to-one.

To assess the impact of the user environment on sessionization heuristics, we considered two important sources of error in heuristic identification of unique users: (1) when a single user accesses the site with multiple IP+agent combinations, and (2) when one IP+agent combination is shared by multiple users. We compared (in the frame-free site) users identified through IP+agent to those identified using cookies.

(1) One user accesses the server with different IP+agent combinations. The most common reason for the use of different IP addresses is dial-up access to an ISP which assigns a dynamic IP address at each new connection. The use of different agents may occur because users upgrade their browser version, or when they use different browsers at different times. Therefore, IP+agent separation will generally construct more "pseudo-users" than there are "cookie-identified users".

Our log contained 5446 different cookies (and hence, by assumption, users). It contained 6849 unique IP addresses, and 8409 unique IP+agent combinations.

77.38% of all users never changed their IP address, and 96.49% never changed their agent. 75.98% never changed their IP+agent combination. Only 6.15% changed their IP+agent combination more than three times, which would represent the expected behavior of a frequent user of the site whose ISP assigns dynamic IP addresses. This may reflect the fact that 10–20% of users of the site are people working at their own desk on campus, or logging in from university computer rooms, i.e., from hard and non-shared IP addresses. However, the remaining users access the site from the outside and can thus be considered representative of users of a broader range of site types.

More importantly, these errors do not propagate to the session level: Fewer than 5% of real sessions contained multiple IP+agent combinations. This suggests that IP+agent could be quite effective if the analysis is done at the session level (without regard to users). However, there could be a significant error if the analysis is to be done at the user level (e.g., for finding patterns among repeat users).

(2) One IP+agent combination is shared by several users. The most common reason for IP address sharing is the use of the same proxy server by different people. Also, the number of market-dominating browsers and versions is small, so that there are many people with identical user agents.

If the visits of different users from one IP+agent are not overlapping in time, and the temporal interval between their activities is long enough, all three heuristics will correctly introduce a session boundary. *Simultaneous* use of the same IP+agent combination by different users is the central problem introduced by the multiple use of IP+agent. Temporally interleaved sessions cannot be segmented correctly by a temporal heuristic. **href** can distinguish the trails of different, simultaneous users only if they access different objects from different referrers. This is not the most usual case: Rather, most users start at a major entry page of the site, which becomes the common referrer for the next access. Moreover, many object requests have an undefined referrer. Some Web servers,

including the IIS used in our test site, leave the referrer undefined under a large number of conditions.

In our log, 86.98% of IP addresses were used by only one, and only 2.8% by more than three users. Shared IP addresses were mainly proxy servers, most of them AOL. The identification performance of IP+agent was better: 92.02% of IP+agent combinations were employed by only one user, and only 1.32% by more than three users. So 8% of IP+agent combinations involved accesses from different users. However, another test of our log showed that *simultaneous* accesses from different users with the same IP+agent combination accounted for only 1% of the log sessions. Thus, our log presents very good conditions for analysis: An IP+agent combination can be equated with one cookie / user in the large majority of cases.

While it would be interesting to split the log and thus experimentally determine the influence of the non-unique IP+agent–user relation on reconstruction quality, the scarcity of these cases, in particular of simultaneous sessions, means that this would probably not produce statistically meaningful results. We have therefore decided to make this the subject of further work with different data. The small percentage of simultaneous sessions from the same IP+agent may be peculiar to the university site we are studying. Commercial sites of e-shops, e-auctions or public administration are accessed by a much more diversified population of users, so that larger numbers of simultaneous sessions from the same IP+agent can be expected. The absence of a reliable assignment of activities to users is not easy to amend: As shown in [SMBN03], the three heuristics show a performance drop of around 20% if the mapping of activities to users is solely based on the IP+agent combination.

6 Impact of Session Reconstruction on Mining Applications

In this section, we investigate the impact of session reconstruction on two applications: the analysis of frequent entry and exit pages, and the recommendation of pages based on previous visitors' usage. We relate the results to those on session reconstruction quality per se, and discuss further work.

6.1 Impact on Entry/Exit Page Analysis

This application is often the first step of Web site analysis, and it is very important for customer-oriented services. It gives insights concerning which pages are first seen; their quality determines whether the user will visit further pages. The exit page is one at which the user abandoned the site; if this leaving is not desired, then page redesign is necessary. Misclassifications of entry and exit pages lead to misinterpretations of user behavior and non-rewarding design efforts, and should therefore be avoided.

To evaluate the performance of the three reactive strategies, we again use the measures of *precision* and *recall*. In particular, let E be the set of entry

pages in the real sessions, and let E_h be the set of pages characterized as entry pages by the heuristic h. Then, the *precision* of h is the ratio of pages correctly classified as entry pages to all pages characterized by the heuristic as such, while *recall* is the ratio of correctly classified entry pages to all real entry pages: $precision(h) = |E \cap E_h|/|E_h|$, and $recall(h) = |E \cap E_h|/|E|$. For exit pages, precision and recall are defined analogously. The results for entry pages are shown in Fig.7. The results for exit pages are virtually identical.

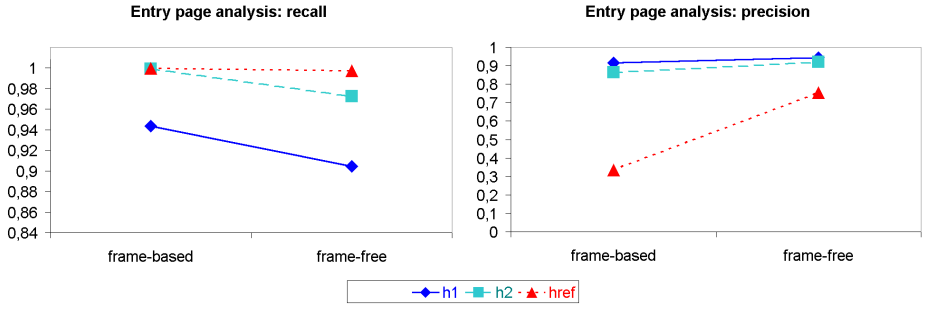


Fig. 7. Recall and precision for entry page analysis by heuristic and site structure.

The different performance of the heuristics can best be explained by relating these values to the number of sessions constructed by them. In particular, **href** constructed very large numbers of sessions, including large numbers of one-page sessions, which turned almost all pages in the site into entry pages. This is likely to find all real entry (exit) pages, creating high recall values. But since it erroneously considers so many pages to be entry (exit) pages, precision is low. The reverse holds for the temporal heuristics, which tend to reconstruct sessions more faithfully.

The breaking up of sessions and the consequent overestimation of the number of sessions was less pronounced in the frame-free site. This was particularly so for **href**. Therefore, fewer incorrect guesses of entries/exits are made, and precision increases. Note that recall performance did not suffer.

In summary, **h1** and **h2** proved to be robust heuristics, both with respect to site structure and in terms of the comparison between their (high) recall and precision values, with a tradeoff between a somewhat higher precision (**h1**), or a somewhat higher recall (**h2**). The precision of **href** is generally lower, and it is strongly affected by the presence of frames. However, it has a very high, and robust, recall performance.

These findings closely parallel the results reported in section 4; witness in particular the structure of results for the frame-free site and for precision values between Figs. 4 and 7. The main difference between these figures is the relative performance of **href** on recall: It reconstructs fewer real sessions correctly than **h1** and **h2** do. This is a consequence of the segmenting behavior of **href**, which

will lead to a high probability that all real entry pages are identified, but will then erroneously cut many of them too soon to capture the remaining real session. This indicates that results on the impact of factors like site structure on the quality of session reconstruction also give information on the impact of these factors on the quality of subsequent mining results.

6.2 Impact on Page Prediction/Recommendation

Recommender systems are another relevant mining application that depends heavily on the correct reconstruction of sessions. Based on co-occurrences of pages found in previous sessions, a new visitor in an ongoing session can be given recommendations for pages that are likely to be interesting to her given the pages visited so far in her ongoing session. One method of determining co-occurrences is based on the clustering of sessions, e.g., [MCS00].

Mobasher, Dai, Luo, and Nakagawa [MDLN02] have proposed a method of evaluating the quality of different clustering methods according to their profiles' predictive power and recommendation quality. In particular, they tested PACT (Profile Aggregations based on Clustering Transactions) against two other methods for clustering and profile creation. PACT was also used here: The transactions are sessions consisting of visits to pageviews p from a set of pageviews P, expressed as vectors of $\langle p, weight \rangle$ pairs. In the current experiments, the weight of a pageview specifies whether it was visited in this session (1) or not (0). The transactions were clustered using k-means, and for each transaction cluster c, the centroid (mean vector) was computed. The weight of a pageview in the mean vector is the proportion of sessions belonging to this cluster in which this pageview was visited. A threshold of 0.7 was then applied,[4] and the result constituted the *profile* pr_c of that cluster: $pr_c = \{\langle p, weight(p, pr_c)\rangle \mid p \in P, weight(p, pr_c) \geq 0.7\}$. This can be represented as a vector in the original space, by using weights of zero for the other pageviews.

Predictive power was measured by the "weighted average visit percentage", WAVP. This allows us to evaluate each profile individually according to the likelihood that a user who visits any page in the profile will visit the rest of the pages in that profile during the same session. It is computed by calculating the average similarity of a profile to each session (their scalar product), averaging over sessions, and normalizing by the sum of weights in the profile [MDLN02].

In [MDLN02], WAVP was used to compare different sets of profiles obtained using different methods of clustering, in order to evaluate the quality of these methods. Here, we used the same method of clustering throughout (PACT), and compare different sets of profiles obtained from different sets of (real or constructed) sessions.

We used data from the frame-free setting, with the standard parameter values (see section 2.2).

In our experiment, we concentrated on the error introduced by session construction. We therefore used a common baseline for testing prediction quality:

[4] The results were similar for other values ≥ 0.5.

the profiles obtained by clustering the set of real sessions. The WAVP values of applying these profiles to these sessions constitute the best possible baseline. Then, the constructed sessions produced by the different heuristics were clustered to obtain heuristic profiles, and WAVP values of applying these profiles to the original set of real sessions were computed.

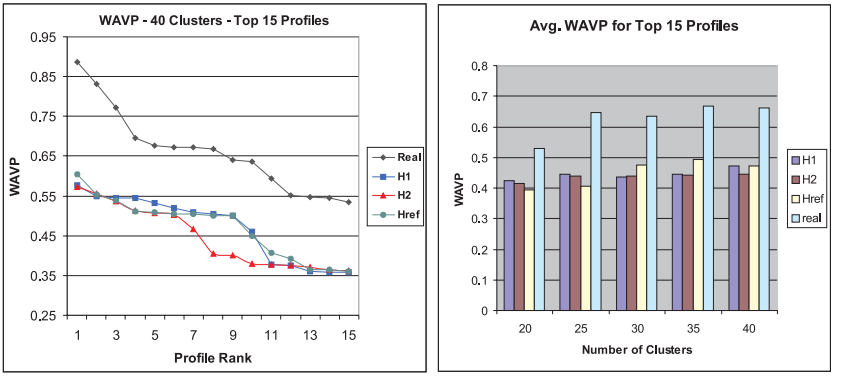


Fig. 8. WAVP of the top 15 profiles: (a) WAVP values at $k = 40$: baseline (real) and profiles built from constructed sessions (**h1, h2, href**). (b) Average WAVP values for different k.

This procedure was repeated for different values of k. The best values for all ways of session (re)construction were obtained for the number of clusters k between 35 and 40. Figure 8 (a) shows the quality of prediction for the top 15 profiles at $k = 40$, ordered by their WAVP value. The differences for lower-ranking profiles are negligible. It shows that the impact of heuristic session reconstruction (the difference between the "real" and "heuristic" WAVPs) is approximately constant for all profile ranks. It also shows that while the difference in performance between heuristics is not large, **href** and **h1** have the highest values and are (near-)identical for most ranks. Figure 8 (b) shows the average WAVP, depending on the number of clusters k. This shows a slight superiority of **href** for $k \geq 30$.

A possible explanation for this result is again the tendency of **href** to construct short sessions, and thus short profiles. These short sessions capture a large percentage of real session entries (see Fig. 7) and probably also the next few pages. In the frame-free site, they also capture the largest percentage of real session content (see Fig. 3). In addition, profile construction selects only the most frequent parts of these short constructed sessions. So many real sessions will contain the requests for the pages in the profiles. Thus, the average similarity and WAVP will be high, in particular for the top ranking profiles that capture the most typical behavior. **href** profiles will tend to generate fewer, but safer, recommendations. In contrast, the prediction quality of **h1** profits from

its rather faithful reconstruction of sessions. (The conclusion regarding **href**, however, will likely not hold up if the site is highly frame-based.)

Taken together, these results indicate that for prediction, **href** and **h1** perform rather well. Future work will include further analysis of recommendations based on the different heuristics for their "usefulness", or the extent to which they include "interesting" items, see [MDLN02] for a methodology.

7 Conclusions

In this study, we have compared the performance of session reconstruction heuristics. We expect that performance is related to the predictive power of the sessions built by the heuristics. Hence, we have compared the quality of predictions for entry/exit pages and of recommendations based on usage profiles. Since session reconstruction heuristics do not necessarily produce the dataset of *real* sessions, we have established and conducted a suite of experiments that compare the datasets of real and of constructed sessions in terms of base statistics, contents and distribution of session lengths.

An essential aspect of our investigation has been the comparative study of the impact of framesets. To this purpose, we have compared our findings on server logs from the frame-based and the frame-free version of a site. Although one cannot claim that any site is representative of all sites in the Web, our experiments indicate that the presence of framesets does not affect all heuristics uniformly. In particular, time-based heuristics are less affected by the presence of framesets, while the referrer heuristic exhibits very poor performance. On the other hand, this same heuristic improves dramatically when reconstructing small sessions in the frame-free site.

An essential application of reactive heuristics is the reconstruction of sessions comprised of page views in multiple servers. Even in the presence of cookies, the synchronization of the cookies or of generated session identifiers is not always possible. Our results indicate that the referrer heuristic, which is per se designed for this kind of application, may perform satisfactorily in frame-free sites with small sessions only. Since the other heuristics exhibit better performance, we intend to investigate how combinations of reactive heuristics perform on the union of logs of multiple servers.

We have studied the predictive power of the constructed sessions for the prediction of entry/exit pages and for recommendations based on usage profiles. Much work in Web usage analysis focuses on the establishment of such profiles, i.e. on descriptive patterns. We intend to elaborate on the impact of session reconstruction on the quality of clusters by establishing an appropriate comparison framework.

References

[BMSW01] Berendt, B., Mobasher, B., Spiliopoulou, M., and Wiltshire, J. 2001. Measuring the accuracy of sessionizers for web usage analysis. *Proceedings of the Workshop on Web Mining, First SIAM International Conference on Data Mining, Chicago, IL.* 7–14.

[BS00] Berendt, B., and Spiliopoulou, M. 2000. Analysis of navigation behaviour in web sites integrating multiple information systems. *The VLDB Journal* **9** 56–75.

[CP95] Catledge, L., and Pitkow, J. 1995. Characterizing browsing behaviors on the world wide web. *Computer Networks and ISDN Systems* **26** 1065–1073.

[CMS99] Cooley, R., Mobasher, B., and Srivastava, J. 1999. Data preparation for mining world wide web browsing patterns. *Journal of Knowledge and Information Systems* **1** 5–32.

[MCS00] Mobasher, B., Cooley, R., and Srivastava, J. 2000. Automatic personalization based on web usage mining. *Communications of the ACM* **43(8)** 142–151.

[MDLN02] Mobasher, B., Dai, H., Luo, T., and Nakagawa, M. 2002. Discovery and evaluation of aggregate usage profiles for Web personalization. *Data Mining and Knowledge Discovery* **6** 61–82.

[PZK01] B. Padmanabhan, Zheng, Z., and Kimbrough, S.O. 2001. Personalization from incomplete data: What you don't know can hurt. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001)*, pages 154–163, San Francisco, CA, August 2001.

[SF99] Spiliopoulou, M., and Faulstich, L.C. 1999. WUM: a tool for Web utilization analysis. *Proceedings EDBT Workshop WebDB'98*, LNCS 1590, Springer, Berlin, Germany. 184–203.

[SMBN03] Spiliopoulou, M., Mobasher, B., Berendt, B., and Nakagawa, M. 2003 A framework for the evaluation of session reconstruction heuristics in Web-usage analysis. *INFORMS Journal on Computing* **15** 171–190.

[W3C99] World Wide Web Committee Web Usage Characterization Activity. 1999. *W3C Working Draft: Web Characterization Terminology & Definitions Sheet.* www.w3.org/1999/05/WCA-terms/

[ZPK03] Zheng, Z., Padmanabhan, B., and Kimbrough, S. 2003. On the existence and significance of data preprocessing biases in Web-usage mining. *INFORMS Journal on Computing* **15** 148–170.

Author Index

Berendt, Bettina 159
Bergholz, André 86

Chi, Ed H. 1

Frías-Martínez, Enrique 66

Geyer-Schulz, Andreas 137

Hahsler, Michael 137
Hay, Birgit 50
Heer, Jeffrey 1
Heskes, Tom 35

Joshi, Neeraj R. 17

Karamcheti, Vijay 66
Kubota, Kazuto 119

Mobasher, Bamshad 159

Nakagawa, Miki 159
Nakase, Akihiko 119

Oyanagi, Shigeru 119

Parthasarathy, Srinivasan 100

Rosien, Adam 1

Shah, Harshit S. 17
Spiliopoulou, Myra 159
Sureka, Ashish 17

Vanhoof, Koen 50

Wets, Geert 50
Wurman, Peter R. 17

Yang, Hui 100
Ypma, Alexander 35