A COLLABORATIVE FILTERING APPROACH TO PREDICT WEB PAGES OF INTEREST FROM NAVIGATION PATTERNS OF PAST USERS WITHIN AN ACADEMIC WEBSITE

by

Denis Lemongew Nkweteyim

BSc. Hons. in Electrical and Electronic Engineering, University College, Cardiff, 1988

MSc. in Digital Communication Systems, Loughborough University of Technology, 1989

Submitted to the Graduate Faculty of

School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2005

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Denis Lemongew Nkweteyim

It was defended on

27 April, 2005

and approved by

Professor Peter Brusilovsky
Department of Information Science and Telecommunications

Professor Jerrold H. May
Joseph M. Katz Graduate School of Business

Professor Paul Munro
Department of Information Science and Telecommunications

Professor Michael B. Spring
Department of Information Science and Telecommunications

Professor Stephen C. Hirtle
Department of Information Science and Telecommunications
Dissertation Director

A COLLABORATIVE FILTERING APPROACH TO PREDICT WEB PAGES OF INTEREST FROM NAVIGATION PATTERNS OF PAST USERS WITHIN AN ACADEMIC WEBSITE

Denis Lemongew Nkweteyim, PhD

University of Pittsburgh, 2005

This dissertation is a simulation study of factors and techniques involved in designing hyperlink recommender systems that recommend to users, web pages that past users with similar navigation behaviors found interesting. The methodology involves identification of pertinent factors or techniques, and for each one, addresses the following questions: (a) room for improvement; (b) better approach, if any; and (c) performance characteristics of the technique in environments that hyperlink recommender systems operate in. The following four problems are addressed:

Web Page Classification. A new metric (PageRank × Inverse Links-to-Word count ratio) is proposed for classifying web pages as content or navigation, to help in the discovery of user navigation behaviors from web user access logs. Results of a small user study suggest that this metric leads to desirable results.

Data Mining. A new apriori algorithm for mining association rules from large databases is proposed. The new algorithm addresses the problem of scaling of the classical apriori algorithm by eliminating an expensive join step, and applying the apriori property to every row of the database. In this study, association rules show the correlation relationships between user navigation behaviors and web pages they find interesting. The new algorithm has better space complexity than the classical one, and better time efficiency under some conditions and comparable time efficiency under other conditions.

Prediction Models for User Interests. We demonstrate that association rules that show the correlation relationships between user navigation patterns and web pages they find interesting can be transformed into collaborative filtering data. We investigate collaborative filtering prediction models based on two approaches for computing prediction scores: using simple averages and weighted averages. Our findings suggest that the weighted averages scheme more accurately computes predictions of user interests than the simple averages scheme does.

Clustering. Clustering techniques are frequently applied in the design of personalization systems. We studied the performance of the CLARANS clustering algorithm in high dimensional space in relation to the PAM and CLARA clustering algorithms. While CLARA had the best time performance, CLARANS resulted in clusters with the lowest intra-cluster dissimilarities, and so was most effective in this regard.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AGNES – Agglomerative NESting
AH – Adaptive Hypermedia
CF – Collaborative filtering
CLARA – Clustering LARge Applications
CLARANS – Clustering Large Applications based on RANdomized Search
DIANA – Divisive ANAlysis
E-commerce – Electronic Commerce
HTML – HyperText Markup Language
HTTP – HyperText Transfer Protocol
IF – Information Filtering
ILW – Inverse Links-to-Word count
IP – Internet Protocol
IR – Information Retrieval
KDD – Knowledge Discovery from Databases
KNN – *k* Nearest Neighbors
LRU – Least Recently Used
LSI – Latent Semantic Indexing
LW –Links-to-Word count
MFR – Maximal Forward Reference
MLP – Multi-layer Perceptron
P3P – Platform for Privacy Preferences
PAM – Partitioning About the Medoid
PR – Page Rank
PR × ILW – Page Rank × Inverse Links-to-Word count
SIS – School of Information Sciences, University of Pittsburgh
SVD – Single Value Decomposition
TFIDF – Term Frequency Inverse Document Frequency
URL – Uniform Resource Locator
W3C – WWW Consortium
WAP – Wireless Applications Protocol
WWW – World Wide Web

ACKNOWLEDGEMENTS

I sincerely thank my advisor Professor Stephen C. Hirtle, for all the help I have had from him during my stay at the University of Pittsburgh. The mentorship, dissertation guidance, and patience given to my work are very greatly appreciated. I also thank all the other members of my dissertation committee for the very useful feedback I got from them in the course of this study.

I heartily thank the Fulbright program for sponsoring the first two years of my stay in the U.S., and the School of Information Sciences, University of Pittsburgh, for giving me the opportunity to complete my studies through the student assistantship program of the school.

I am forever grateful to the University of Buea in my home country of Cameroon, and especially the Vice-Chancellor, Dr. Dorothy L. Njeuma, for endorsing my application to pursue my doctoral studies, and for guaranteeing my return to employment at the end of my studies.

My mentor for over 20 years, Mark Hammond of Huntington, W.V., his family, and Grandma Dorothy Hammond have been a source of great support and encouragement for me and my family, and I thank them sincerely for their friendship and kindness.

I have also received invaluable support from my church, the Bellefield Presbyterian Church of Pittsburgh, PA, and I would like to thank them for all the love and support they have given me and my family.

My parents, Thomas and Matilda Nkweteyim, and all my siblings: Emmanuel, Pat, Odilia, Joan, Vivian, Joyce, and Clifford, and my uncle, Philip Forlemu and his family, have all been pillars of support for me, and I thank them very sincerely.

Finally, my wife Beatrice, and our three daughters Daisy, Raisa, and Whitney, have given me all the love and support I could ever ask for, and I thank them for being so patient with me.

Dedicated to sweet memories of
My mother, Matilda Yimnai Nkweteyim, who lost
Her battle with cancer just before I completed this report,
And my friend and brother-in-law, François Temeching.

1. INTRODUCTION

The need for web navigation aids is a paradox of the new information age: at the same time that technology is meeting the user's thirst for access to information anytime and anywhere by making the retrieval, production and distribution of information easy and affordable, additional technology is required to spare the user from being inundated with too much of this information, so that he gets just what he needs with little or no distracting information. The problem the user faces is how to divide his attention between the numerous sources of information available to him, much of it irrelevant or inaccurate (i.e., data smog (Shenk, 1997)). As Herbert Simon put it: "What information consumes is rather obvious: it consumes the attention of its recipients. Hence a wealth of information creates a poverty of attention, and a need to allocate that attention efficiently among the overabundance of information sources that might consume it" (as quoted by Varian (1995)). There is therefore, a need to guide the user to relevant information without causing him to navigate through noisy, but equally attention-seeking information. This need is particularly necessary, as more and more of the available information, much of it on the Internet[1], is unstructured at the same time that the web user base, comprising largely of ordinary computer users, and not the expert programmers that early computer users were, is expanding.

Users are inundated with information not only when they actively seek it as in web browsing, but also when they do not, as evidenced by the amount of unwanted email that they receive in their mailboxes, or spy ware that they may not even be aware is installed on their computers. The combination of these factors means that more and more people many of them not able to effectively specify their information needs, are now required to manage more and more information, if they are to be able to access the information they really need and ignore what they do not need. Marchionini (1995) summarizes the dilemma faced by users in the following way:

> …we travel a narrow road between our goals with a sea of seductive information to distract us on one side and a spiraling abyss of confusion and information overload on the other. Technology increases the rate at which

[1] It has been estimated that (1) print, film, magnetic, and optical storage media produced about 5 exabytes of new information in 2002 with 92% of the new information stored on magnetic media, mostly in hard disks; (2) the amount of new information stored on paper, film, magnetic, and optical media doubled between 1999 and 2002; and (3) information flows through electronic channels—telephone, radio, TV, and the Internet—contained almost 18 exabytes of new information in 2002, with 98% of this sent and received in telephone calls - including both voice and data. Sources of this information flow include the World Wide Web (170 terabytes, equivalent to seventeen times the size of the Library of Congress print collections; instant messaging (274 terabytes a year); email (400,000 terabytes a year), and increasingly, P2P file exchanges (Lyman & Varian, 2004).

> we are able to travel towards our goals, but it also increases the scope and peril of the two sides.

The terms *information overload* and *lost in hyperspace* (Thimbleby, Jones, & Theng, 1997) are used to describe problems that users face as they try to manage the information they have to deal with. Alvin Toffler coined the term *information overload* (also known as *cognitive overload*), which “refers to the state of having *too much* information to make a decision or remain informed about a topic” (Wikipedia, 2004), which often exceed their cognitive capacities. The overload is the consequence of the need for additional effort and concentration if users are to perform several tasks at the same time. *Lost in hyperspace* describes the tendency of users to lose their way because of the nonlinear nature of browsing in a hypertext environment. Nielsen (1995) described the *lost in hyperspace* phenomenon as follows:

> When users move around a large information space as much as they do in hypertext, there is a real risk that they may become disorientated or have trouble finding the information they need...Even in [a] small document, which could be read in one hour, users experienced the 'lost in hyperspace' phenomenon as exemplified by the following user comment: 'I soon realized that if I did not read something when I stumbled across it, then I would not be able to find it later.

Nentwich (2002) described the same phenomenon as follows:

> When reading a hypertext, following links just as they seem interesting may soon lead to the reader's perception to be somewhat "lost", i.e. that s/he does not know any more where exactly one "stands" in the hypertext structure, how far it is until the end, how much of the whole is already read etc. This is due to the fact that – in contrast to a linear paper – we have much less hints with regard to the whole.

Traditionally, web users have made use of a number of technologies to help them access information on the web. These include the use of search engines, human-organized directory browsing like Yahoo and membership to distribution lists. These systems have shortcomings though. Human-organized directories are expensive to create and update, and coverage of topics is not exhaustive. For search engines, there are two main problems (Belkin, 2000). The first problem is that most users do not know exactly how to phrase queries to retrieve the documents that they really need. The second problem is that most users are not aware of the underlying operation of search engines, and the nature of the vocabulary (keyword-based, or the use of a controlled vocabulary) that the search engine requires in user queries. Hence, in spite of the usefulness of search engines, the results returned from queries are usually noisy, comprising of both relevant and irrelevant documents. Finally, distribution lists have the problem of attempting to serve the general interests of a group of users, and not the particular needs of the individual user (Malone, Grant, Turbak, Brobst, & Cohen, 1987).

Personalization systems can alleviate both the *information overload* and *lost in hyperspace* problems by causing a website to be more responsive to the unique and individual needs of each user (Cingil, Dogac, & Azgin, 2000). Personalization involves the creation of user profiles and tailoring the information received by the user to his profile. The following remark by Bezos, the CEO of Amazon.com, probably the most famous e-commerce site, illustrates well, the importance of website personalization: “If I have 10 million visitors to my website, I should have 10 million websites for my visitors” (as quoted by Riedl, 2001).

Recommender systems are a category of personalization systems that make individualized recommendations to users during a live interaction with the system (Sarwar, Karypis, Konstan, & Riedl, 2001). The majority of recommender systems in use today are in the domain of e-commerce. Most of them require some input from users in the creation of their profiles.

There is however, need for recommender systems in domains other than e-commerce. In some domains like web navigation, it is unrealistic to expect users to continually provide user profiling or document ratings information to the system as their interests change, or as they navigate a website. There is therefore a need to develop recommender systems that do not require the user to explicitly provide information on their interests.

This project is a study of factors and approaches used in the design of hyperlink recommender systems, based on implicitly acquired user profiling data, in the domain of web navigation for a user visiting a website. The work involves an analysis of some existing approaches and technologies currently used in recommender systems design, it proposes possible improvements to some existing approaches, and proposes some new ones. All of the proposals were implemented and evaluated to determine their benefits.

The underlying assumption in the project was that hyperlink recommendations to a user of a website can be based on the web pages that past users with similar browsing behavior of that web site found interesting. Hence, by discovering patterns of past user navigation behaviors from user access logs, a hyperlink recommendation system can be constructed for current users.

The rest of this report is divided into the following sections. Section 2 is a literature survey of issues related to recommender system design, and application areas. Section 3 defines the problem that this study addresses. Section 4 presents the implementations of simulations used to test the hypotheses that were tested in the study. Section 5 presents the findings and discussion of results, Section 6 the conclusions and an overview of future directions of the study.

2. RECOMMENDER SYSTEMS: BACKGROUND

This section presents a survey of recommender systems technology. The section is discussed under the following headings. Section 2.1 defines recommender systems, highlights differences from other systems that serve as aids to users of large information spaces, and describes the components of a typical recommender system. Section 2.2 presents a framework for understanding recommender systems. Section 2.3 discusses performance related issues in recommender system design. Section 2.4 discusses privacy issues in recommender system design, highlighting the tension between users' desire for privacy and designers' desire for more information about users.

2.1. What is a Recommender System

Resnick and Varian (1997) proposed the term *recommender system* to represent a system that takes people's recommendations of items as inputs and uses these recommendations as a basis for making recommendations to other people. Today, this definition only applies to one class of recommender systems: collaborative filtering systems. A more inclusive definition today is "any system that produces individualized recommendations as output, or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options" Burke (2002). Recommender systems may be regarded as tools that help users in their information-seeking task[2]. They are a specialized kind of personalization systems that use dynamic user profiling to generate personalized recommendations in a live interaction with the system. This contrasts with other personalization systems in which the user profile is static, and for the same user action, the user always receives the same information personalized to that profile. Recommender system designs employ a wide range of techniques drawn from various domains, including information retrieval (IR), user modeling, and machine learning. Application areas of recommender technology have also evolved from the traditional domains of music, movie and news, to others, including recommendation of products at e-commerce sites, restaurant recommendation, job recommendations, recommendation of relevant research

[2] Marchionini (1995) defined information seeking as "any process by which people purposefully engage in a process to change the state of their knowledge as they seek to obtain information from automated information systems." The information sought may or may not be available in the information system.

literature, TV viewing recommendations, and recommendation of hyperlinks to help users more easily navigate websites.

Recommender systems differ from other information management systems in that recommendations are tailored to the needs of the individual user, and not to the needs of a group of users as in a distribution list, for example. Another characteristic feature that distinguishes recommender systems from other information seeking tools is the fact that recommendations are computed during a live interaction with the system (Sarwar, Karypis, Konstan, & Riedl, 2000b; Sarwar et al., 2001). Latency is thus an important consideration in recommender system design; users will be put off if it takes too long for the system to generate recommendations.

Figure 1 is a sketch of the components of a recommendation system. The user interacts with the system, and this interaction, coupled with the user profile, leads to generation of individualized recommendations, and updating of his profile.

The User Profile. As can be seen from the figure, there is a distinction between the user who has information needs to satisfy, and the user profile, which is a machine representation of what the system believes the user's needs to be. The major difficulty in generating relevant recommendations is the potential for mismatch between real user interests and the user profile. A major goal in recommender system design therefore, is to model the user well enough for the system to be able to generate relevant recommendations and at the same time filter out irrelevant ones.

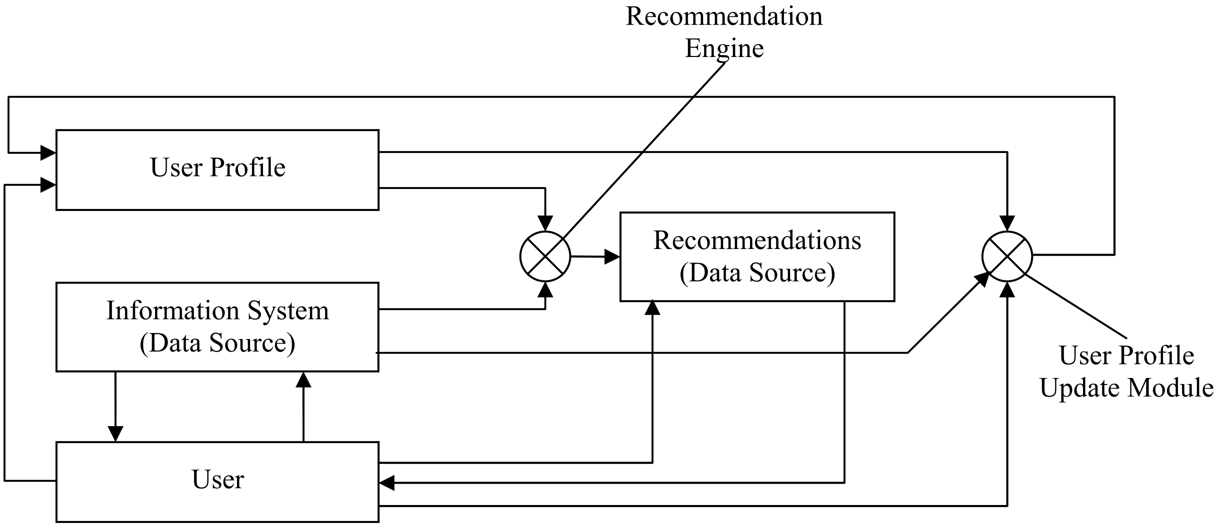

Recommendation Engine
User Profile
Recommendations (Data Source)
Information System (Data Source)
User Profile Update Module
User

Figure 1. Components of a recommender system. Arrow directions indicate the sources and destinations of data used by the system, and of user actions.

When a user starts using the system, his profile is initialized to one of the following states: a predetermined state, the previous state for that user if the system maintains a long-term record of user profiles, or a state explicitly specified by the user. The user profile may then be updated based on his interaction with the system.

User. The user is the central component of the recommender system. The user receives data from the information system through an interface. This data could be static (for example, the web pages in a website) or dynamic in the form of recommendations that the system makes. The user in turn interacts with these data, and these interactions, together with his current profile, determine what recommendations are made, and how his profile is updated.

Recommendation Engine. The recommendation engine examines the user profile and makes recommendation of items from the information system, based on the profile.

2.2. Framework for Understanding Recommender Systems

This section presents a taxonomic framework that depicts different ways to view and classify recommender systems. The framework (Figure 2) presents recommender systems with respect to six main parameters, namely the target of the recommendations, type of recommender system, user profiling issues, associated technologies, recommender system in the context of other related technologies (IR, information filtering (IF), and adaptive hypermedia (AH)), and major application domains. Each major category is further divided into sub categories. It should be noted that this taxonomy does not place recommender systems firmly in one group or another. It is simply a tool to help explain what recommender systems are. As a matter of fact, several recommender systems share characteristics that belong to several of the groupings that are identified in the taxonomy.

2.2.1. Target of Recommendation

Most recommender systems are designed to make recommendations based on the needs of individual users. Recently however, there has been some work on group recommendations. For example, PolyLens (O'Connor, Cosley, Konstan, & Riedl, 2001) is designed to make movie recommendations to a group of users, as opposed to individual users; MUSICFX (McCarthy & Anagnost, 1998) allows members of a fitness center to influence, though not directly control, the selection of music in a fitness center based on

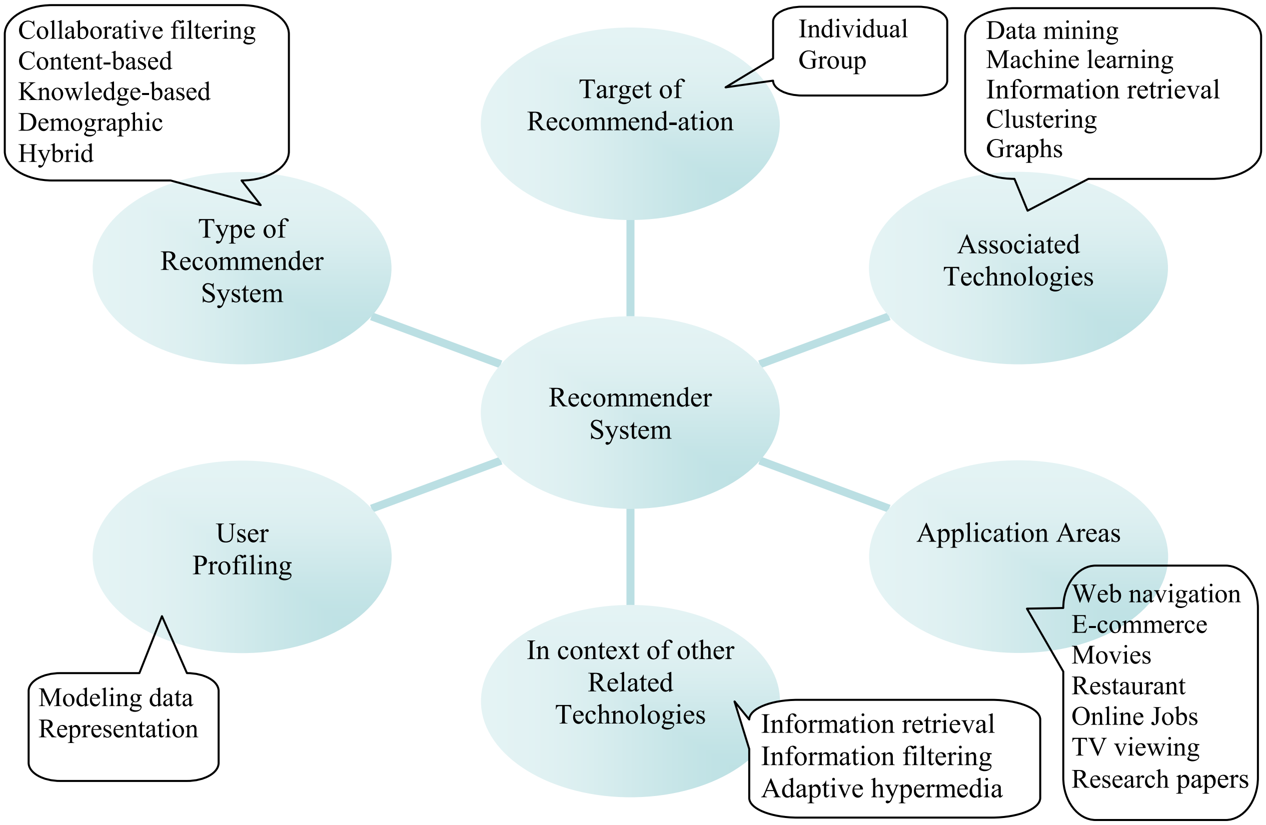


Figure 2. Recommender system taxomomy.

the music preferences of members of the center, and which of those members are currently using the center. Group recommendation makes sense in situations where participants would prefer to act in a group, like watching a movie together.

2.2.2. Type of Recommender System

Collaborative filtering and content-based recommenders are the most widely available categories of recommender systems. Some more recent systems use a knowledge-based approach to generating recommendations. Recommender systems that make use of demographic information have also been proposed. Finally, hybrid systems, which make use of more than one recommendation technique, are common. This section reviews recommender systems based on this categorization[3].

[3] Other types of recommender systems that do not fit into this categorization exist. For example, Shani, Brafman, and Heckerman (2002) describe an approach to recommender systems design based on Markov decision processes. We limit our discussion to the four categories mentioned here.

2.2.2.1. Collaborative Filtering (CF) Systems

Collaborative recommenders or collaborative filtering systems suggest new items or predict the utility of a certain item for a particular user based on the user's previous likings and the opinions of other like-minded users (Sarwar et al., 2001). The process of generating recommendations by collaborative filtering is summarized in Figure 3.

The system comprises a database of m users $U = \{u_1,u_2, ...,u_m\}$ and n items $I = \{i_1,i_2, ...,i_n\}$. Each user u_i has rated a number of items I_{ui} either explicitly or implicitly from the set I. The aim of collaborative filtering for the active user u_a, is either to predict the *likeness* score for item $i_j \notin I_{ua}$, where I_{ua} is the set of items already rated by u_a, or to recommend a list of N items, $I_r \subset I$ and $I_r \cap I_{ua} = \phi$. By comparing the active user's ratings to those of other users using some similarity measure, the system determines users who are most similar to the active user, and makes predictions or recommendations based on items that other similar users have previously rated highly. The profile of a user comprises a vector of items ratings, with the ratings being binary or real-valued.

Collaborative recommenders may be *memory-based* or *model-based* (Sarwar et al., 2001). A memory-based collaborative filtering algorithm uses a *nearest-neighbor* algorithm that determines a set of neighboring users who have similar ratings characteristics as the active user, and combines their preferences to determine a prediction or list of recommendations for the active user. Most current collaborative filtering systems are memory-based, and work as shown in Figure 3. They are more efficient than model-based ones if the ratings database is not too large. Speed however, becomes a bottleneck if the ratings database is large because of a much larger number of nearest neighbor computations, and in such situations, model-based recommenders are an attractive alternative.

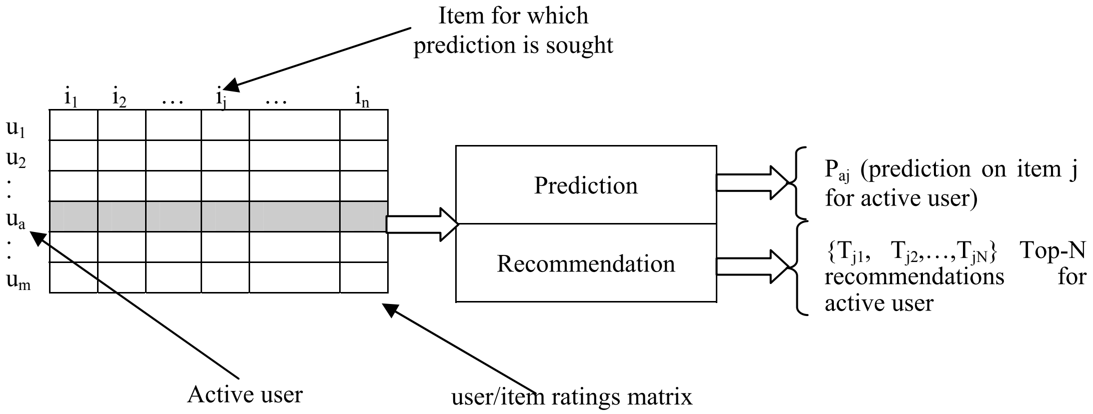

Item for which prediction is sought
i1 i2 … ij … in
u1
u2
:
ua
:
um
Prediction
Recommendation
Paj (prediction on item j for active user)
{Tj1, Tj2,…,TjN} Top-N recommendations for active user
Active user
user/item ratings matrix

Figure 3. The Collaborative Filtering Process. Source: Sarwar et al. (2001).

Model-based collaborative recommenders do not use the user-item database directly to make recommendations or predictions. Instead, they use a machine learning approach to build a model of user ratings, and make future predictions and recommendations based on this model. For example, *item-based* collaborative filtering (Sarwar et al., 2001), is a model-based approach to predicting a user's likeness for an item that depends on the following intuition: a user would be interested in items that he liked before and would tend to avoid items that he did not like before. The technique first analyzes the user-item ratings matrix to identify the similarity between items, where the similarity between two items is computed by a function that compares the ratings given to the items by users who co-rated them with the current user. Once the items most similar to the one the user is interested in are found, the prediction score for the item of interest can be computed based on the weighted sum of the user's ratings of the similar items, or using a regression model. Item-based techniques are useful in large databases where the set of items is relatively static as in a typical e-commerce scenario, and item-item similarities can be computed off-line, which means considerably less time spent generating recommendations during a live interaction with the system. But this requires an $O(\mathrm{n}^2)$ increase in space requirements. The space complexity could be minimized if only the k ($k<n$) most similar items to the item of interest are considered.

Several factors make classical collaborative filtering systems good for making recommendations. First, they are independent of machine-readable representation of objects being recommended and the associated errors that such representations entail. This makes them good choices for cross-genre recommendation. For example, in music and movie recommendations, where preferences are determined by personal tastes, a system that makes recommendations based solely on attributes like actor and producer names, film or music genre, etc., will miss these personal tastes. Second, there is no need for domain knowledge to make recommendations; hence they avoid a possibly difficult knowledge engineering problem. Third, system effectiveness improves over time as more items are rated, increasing the number of similar users whose ratings can be compared with those of the current user.

There are also a number of problems associated with the collaborative filtering technique.

The ramp-up problem (Burke, 2001) comprises two related problems: the *new user* or *cold-start problem* (Burke, 2001) and *the new item*, also called *early rater* problem (Burke, 2001; Claypool et al., 1999). The problem with the new user is that he has made too few or no recommendations, and any attempt to make predictions or recommendations for such a user by comparing his ratings with those of others who have made similar recommendations as him, is error prone. A new item is a problem because it may have very few ratings, and the early raters gain little because their ratings only go to improve recommendations for other users. There is thus little incentive for users to rate new items.

The sparsity problem (Konstan et al., 1997) arises from the reliance of collaborative filtering systems on the overlap between different users' ratings of objects to determine similarity between users. The problem arises if there are several items that have been rated by only a few users, even if these users have each rated several other items, but only few of the ratings overlap between them, resulting in low ratings densities for these items. In such situations, the system will not be able to make reliable recommendations because there are not enough similar users available.

Scalability is a major problem with collaborative filtering systems (Ungar & Foster, 1998). The improvement in system accuracy with increasing ratings comes at a price: as the number of users and items in the database increases, so too does the number of nearest-neighbor computations required for the determination of similar users, and since recommendations are made in real time, these systems suffer as a result.

The *gray sheep* problem (Burke, 2002; Claypool et al., 1999) occurs in collaborative recommenders when there are users—the gray sheep—whose profiles fall on the border between existing classes of users, making it hard to determine what recommendations to make for these users.

Collaborative recommenders also suffer from the *portfolio effect* (Towle & Quin, 2002). This occurs in domains where it is desirable not to recommend an item that the user already has, for example, news items that the user has already read. A collaborative filtering system will continue to recommend very similar news stories if they had not been recommended to the user before.

The *synonymy* problem (Sarwar et al., 2000b) arises from the lack of any form of semantic interpretation by collaborative filtering systems. Hence, items that are very similar are not treated as such when recommendations are made. For example, two almost identical concepts like *recycled paper pad* and *recycled memo pad* are not considered as highly similar, namely, *recycled office products*[4]. They are treated as distinct items, and both may be recommended to the same user. A related problem resulting from the lack of semantic interpretation in collaborative filtering and related technologies is the *polysemy* problem, which refers to the inability of the system to recognize that the same term may have different meanings, depending on the context of use, and erroneously failing to recommend them in these different contexts.

Other problems with collaborative recommendation systems include the subjective nature of the ratings, which is prone to user bias, and the fact that user profiles may easily become outdated if users' interests change and they do not use the system regularly enough for their profiles to be updated.

2.2.2.2. Content-based Recommenders

In content-based recommendation, features of the recommendable objects are used to define them (Balabanovic & Shoham, 1997a; Claypool et al., 1999). For example, the words in a document could be used as the defining features of that document. The user profile in such systems is based on how these features are rated for the user. Typically, the system watches the user as he interacts with the information space, updates his profile accordingly, and recommends new pages that correlate with his profile.

[4] This example is taken from Sarwar et al. (2000b).

Content-based recommenders typically represent documents similarly to IR techniques, for example, making use of indexing and the vector space[5] model to represent documents and the user. The model represents objects as unique fixed-length vectors of features, usually words of a document, weighted by some criteria. Recommendations are made by comparing the user profile with documents present in the system, and only those documents that are very similar to the user profile are recommended. This is analogous to comparing a user query with a document collection in a search task, and returning the documents that are most similar to the query, as the result of the search.

There are a number of advantages in content-based recommendation. First, recommendation is based on features of the document and not other users' biased opinions about the document. Second, it is possible to generate explanations for why a document was recommended based on a comparison of the user profile and the features of the document that contribute to the recommendations. Third, the sparsity problem is avoided since modeling information is present in the features of the document and does not need to be provided by other users of the system.

There are also a number of problems associated with content-based recommender systems. First, the vector representation for documents and user profiles, also called the *bag-of-words* model, does not consider the meanings or order of words in a document as being important. The effect of this is to miss any semantic or syntactic interpretation that could be sources of useful information.

Another problem with content-based recommenders is the *start-up* problem. These recommenders require a large number of training items before they can correctly rate and classify them.

A third problem arises from the fact that content-based recommenders are feature-based: the recommenders will only give good results if the features that are selected to represent documents and user profiles are rich enough to correctly represent them. This is not always possible, for example, in the movie domain, where only simple features like names of actors and producers, and expert commentaries, are used to describe the movies. The problem is that natural language (and the corresponding computer representation) is not rich enough to capture all relevant features, like taste.

Content-based recommenders also suffer from the portfolio effect. In domains like news recommendation, it is possible to filter out news items that are very similar to those that have been presented before, but the problem arises with follow-up news items that may be very similar to a previous one, but at the same time, present some new and important facts. Some systems (e.g., Billsus & Pazzani, 1999b, 2000) attempt to overcome the problem by using a high threshold value to filter similar news items.

The user profiles of collaborative and content-based recommenders as described above, are long-term generalizations about the user, and can be updated as the system learns more about the user or as the user preferences change. These systems suffer the ramp-up problem in one way or another, for which reason casual users will not benefit fully from them. On the other hand, long-term users who are willing to allow the system

[5] The vector space model is examined in Section 2.2.4.3.

to learn their profiles over time may reap significant benefits from the recommendations they receive. A converse problem to the ramp-up problem faced by collaborative filtering and content-based recommenders is the *stability vs. plasticity problem* for user profile-learning algorithms (Burke, 2002). Once a user profile has been established, it is hard for the learner to adapt to changes in the profile, until a sufficiently large number of newer ratings have been collected. This is the problem of adapting to *concept drift*, discussed in Section 2.2.3.2.

2.2.2.3. Knowledge-based Recommenders

Knowledge-based recommenders like Entrée (Burke, 2001, 2002) and Quickstep (S. Middleton, De Roure, & Shadbolt, 2001) make recommendations based on inferences about a user's needs and preferences. These systems have knowledge about how a particular item meets a particular user needs and can reason about a need and possible recommendations. The user profile could be any knowledge structure that can make these inferences.

Strong points of knowledge-based recommenders include (1) the absence of a start up problem since ratings are not required, and (2) the fact that they do not place additional cognitive load in the form of a requirement for user ratings, which makes them ideal for the casual user.

Knowledge-based recommenders suffer from two main problems: model construction may require a significant amount of knowledge engineering effort, and second, the models are more difficult to adapt or extend. The difficulty of adapting and extending the model is particularly significant in noisy environments like the web, where it is more difficult to continually handcraft a representative knowledge base for these changing data (Zukerman & Albrecht, 2001).

2.2.2.4. Demographic Recommenders

Demographic recommenders categorize users based on personal attributes, which are used to classify them as belonging to a given stereotypical group, usually for marketing purposes. For example, indicators such as zip code or annual salary could be useful in determining which demographic stereotype a user belongs to. The recommendations that a user receives are the same that other users of the same demographic class receive.

The LifeStyle Finder (Krulwich, 1997), for example, used demographic groups from marketing research to suggest a range of products and services depending on which one of 62 demographic groups the user belonged to; a survey was used to gather the data for user categorization. Pazzani (1999) on the other hand, used machine learning to extract demographic features like ethnicity and user home town from their home pages, and these features were used to predict which restaurants they would like, and which they would not like.

One advantage of demographic recommenders is that they lack the *new user* problem since they do not require user ratings. Second, as with collaborative recommenders, demographic recommenders can make cross-genre recommendations.

These recommenders do have a number of problems though. The "gray sheep" problem is one of them, and arises for users whose characteristics fall somewhere between those of existing groups of users. A much more serious problem has to do with privacy concerns: users are not likely to disclose important demographic information that they consider private, for example their salaries, for fear of misuse. Like collaborative and content-based recommenders, demographic recommenders suffer the stability/plasticity problems.

2.2.2.5. Hybrid Recommender Systems

Hybrid recommenders Burke (2002) combine more than one recommendation technique to gain better performance with fewer of the drawbacks of any one technique used alone. This approach has been used in a number of recommender systems, including Fab (Balabanovic & Shoham, 1997b) and Profbuilder (Ahmad, 1999) for web navigation, ClixSmart (Smyth & Cotter, 2000) and TV Scout (Baudisch & Brueckner, 2002) for TV viewing recommendation, and Quickstep (E. S. Middleton, Shadbolt, & De Roure, May 2002; S. Middleton et al., 2001) for research paper recommendation.

2.2.3. User Profiling

A user profile is a representation of the user's current interests, and forms the basis for the generation of recommendations; fairly accurate user profiling leads to relevant recommendations, while poor user profiling results in poor recommendations. As users interact with recommender systems, their profiles, are generated, stored, and updated, based on their interaction with the system. User profiling is a challenging aspect of personalization systems due mainly to the fact that the process is very uncertain. This uncertainty results from the fact that the information available to model user behavior is usually limited or noisy, making it hard to infer non-trivial assumptions about the user. This section discusses the following important considerations in the construction of user profiles in personalization systems: approach used (Section 2.2.3.1); *concept drift* (Section 2.2.3.2); methods of acquisition of profiling data (Section 2.2.3.3); useful indicators of user interests (Section 2.2.3.4); sources of data used to construct user profiles implicitly (Section 2.2.3.5); and representation of user profile (Section 2.2.3.6).

2.2.3.1. Approaches in User Profile Construction

There are two main approaches to constructing a user model for recommender systems: knowledge-based and behavior-based. Knowledge-based models are pre-programmed knowledge representation structures that are constructed by analyzing a representative sample of instances of the problem and dynamically matching users to the closest model. Behavior-based models on the other hand, are statistical models that use users' behaviors as the basis of recommendation. Rather than rely on a preprogrammed model, the knowledge about the user is acquired incrementally by making inferences from user actions. Construction of behavior-based models typically involves applying some machine-learning algorithm to users' past behaviors in order to predict their future behavior. Two main approaches are used to learn the user model in behavior-based systems: content-based and collaborative approaches. The content-based approach assumes that people will behave similarly under similar circumstances. Hence, if a user liked an item before, similar items may be recommended for that user in future. The collaborative approach assumes that like-minded people exhibit similar behaviors under similar circumstances. Hence, prediction of a user's preference for an item is based on the aggregate behaviors of other similar users.

2.2.3.2. Concept Drift

The term *concept drift* is used to describe the continuous change in a user's interests and behaviors as he interacts with an information system. Learning algorithms that track a user's profile as he interacts with the system need to adapt to these changes quickly, so that the profile can be updated. Research on machine learning of the user model has traditionally been based on the assumption that the user is engaged in repeated and unchanging behavior, which the learning algorithm can learn (Maes, 1994). The consequence of this has been that users are placed under static classes that may meet the characteristics of groups of users but that are not flexible enough to the changing characteristics of individual users. Because of the dynamic nature of the user profile in recommender systems, good user profiling needs to allow for dynamic profile update as the user's interests change.

Webb et al. (2001) presented a number of approaches that may be used to alleviate the problem of concept drift in personalization systems. They included the following:

Placing Less Weight on Older Observations. The idea here is to make more recent user activity play a greater role in determining the current user interests. But this is done at the risk of losing information about interests that are long-term but occur only sporadically. For example, a user in a news recommendation system may be interested in volcanic mountain eruptions, but because mountains only seldom erupt, the system may fail to recommend a news item on the next mountain eruption to the user because by the time it occurs, the profile indicator for mountain eruptions may have decayed below the recommendable threshold.

Use of Windowing Techniques. Using this approach, user modeling data is limited to a time window, which is considered to represent the user's interests at that period in time.

Use of Dual User Models. This approach makes use of long- and short-term user models. The short-term model, trained with recent data, is initially consulted and if accuracy is low, the long-term model is used instead. Billsus and Pazzani (1999a; 1999b) for example, used a nearest-neighbor short-term user model to adapt rapidly to changing user interests, and a Bayesian classifier long term model to track long term user interests.

2.2.3.3. Acquisition of User Profiling Data

Data that are used to construct user profiles may be acquired *explicitly*, *implicitly*, or using a combination of both explicit and implicit approaches (Hanani, Shapira, & Shoval, 2001). Systems that acquire user modeling data explicitly require users to provide personal information about their interests, likes and dislikes. Acquisition of this information may take a variety of forms, including: form filling to indicate interests, selection of one of several predefined profiles, selection of a set of terms, with the possibility to weight terms based on their relative importance, and rating of items in collaborative filtering (Hanani et al., 2001). Other methods to explicitly acquire the user profile include user interviews, and stereotypical inferences (M. Bauer, Pohl, & Webb, 1997).

The main advantage of explicitly acquiring the user profile is that resulting profile would be reasonably accurate if the user succeeds in choosing the right terms to describe himself. Another advantage is that the system output would be predictable, as the user can usually guess why he is receiving the recommendations he gets, knowing the way he constructed his profile.

There are a few disadvantages in acquiring the user profile explicitly though. First, if users are required to update their profiles each time their interests change, that may place a large cognitive load on them, defeating the original objective of building a system that reduces information load. Furthermore, in systems involving navigation in large information spaces, the additional cognitive load my end up disorienting the user, confounding the *lost in hyperspace* problem. Some of these problems can be alleviated by requiring minimal input from the user, perhaps in the form of a single click of the mouse to rate the quality of the system's output.

Another problem related to explicitly acquiring the user profile is the possibility that the user may not be willing to cooperate, especially if he cannot immediately see the benefits of making personal information available to the system (Mulvenna, Anand, & Büchner, 2000), and also because of privacy concerns. And even if the user is willing to provide modeling data, he may not be able to correctly specify information that correctly defines his profile. Reasons for this range from the failure to know the correct vocabulary to use to correctly describe himself to the system, to, in systems in which a controlled vocabulary is provided for user profile definition, misinterpretation by the user of the meanings of the vocabulary terms (Belkin, 2000).

Systems that do not acquire user profiling data explicitly do so implicitly, through a machine learning procedure. In such systems, the user is not required to be actively involved in the task of acquiring his profile. Instead, the system watches the user's actions and infers a model or profile for the user based on those actions. The underlying assumption in such systems is that by analyzing the behaviors of users, it is possible to find clues on how to personalize the system to their needs.

The main disadvantage of the implicit model is the greater uncertainty of the inferences made simply by observing user behaviors. However, there are a number of advantages. First, the approach is unobtrusive. Second, relieving the user of the burden of building and maintaining a profile is an attractive option. Third, the amount of modeling information that is collected can be very large, and that means a statistically accurate user model can be constructed as a result.

2.2.3.4. Implicit Indicators of User Interests

A number of studies have been performed to determine factors that could serve as implicit indicators of users' interests in documents they encounter as they browse the web. Of these, the amount of time the user spends reading a document has been the most studied. This, and other implicit interest indicators are presented in this section.

Reading Time. Reading time is the time spent reading a page. For web server log data, the reading time for a page can be determined by the difference in the time between the request of the page and a subsequent page request. Reading time can only be an approximate measure, as it depends on network traffic, which is not guaranteed to be steady, and also on actual user action on the document, which may be different from reading the document, but which the system cannot be aware of. Nevertheless, for a statistically large sample, the measured reading time should give an accurate assessment of the actual reading time.

Morita and Shinoda (1994) in a tightly controlled experiment, studied the time used by users of NetNews articles and made the following findings:

- Users spent a long time reading articles they rated as interesting
- Users did not spend a long time to read articles they rated as not interesting
- A very low correlation between the length of an article and the time to read it, suggesting that not all articles were read completely; after reading the first few lines of an article, the user probably abandoned it if he found the article uninteresting
- Readability (i.e. denseness, determined by number of characters per line, or per article, or number of blank lines) of an article was not an important factor affecting its reading time
- A very low correlation between the number of unread articles and reading time, that is, users did not rush over articles even if they had a backlog.

Konstan et al. (1997) also reported that in a loosely controlled, natural setting, the time spent reading USENET articles was a good indicator of user preference, irrespective of the length of the article.

In the domain of technical literature, Kim, Oard, and Romanik (2001) found that users spent more time reading interesting material than they spent reading less interesting ones. They also concluded that the interpretation of reading time should depend on the type of document, with abstracts of journal articles taking longer to read than the times taken to read news articles reported by Konstan et al. (1997) and Morita and

Shinoda (1994). Kim et al. (2001) reported that a combination of reading time and printing of a document could provide a more useful indicator of user interest in a document than reading time alone.

Claypool, Le, Waseda, and Brown (2001) monitored potential interests indicators as users browsed the pages of a college website, and found that the time spent on a page, the amount of scrolling on a page, and a combination of time and scrolling had a strong correlation with explicit user ratings for those pages. The number of mouse clicks was not a good indicator of interest on pages. However, the time spent moving the mouse on a page was a good indicator of low recommendations (little time spent moving the mouse if the page was not interesting), but could not distinguish among high interest pages.

Kelly and Belkin (2001) came up with an apparently contradictory finding to the notion that the time users spend reading a document is an indicator of the user's interest in that document. The authors found that users spent significantly less time reading documents that they classified as belonging to one of six topics, than did users who rated documents as interesting in Morita and Shinoda (1994). In addition, they found no significant difference in the time spent by users reading both documents that belonged to, and documents that do not belong to the topic under consideration. But as the authors pointed out, their results did not really contradict earlier findings; they simply suggested that the earlier findings apply to a narrower scope than previously thought. Their work differed from earlier works on reading time in a number of ways: it was carried out in the IR domain where the tasks were unfamiliar and were not necessarily interesting to the user as was the case in earlier studies where the users rated items in newsgroups that they were interested in; also, in Kelly and Belkin (2001), users were given less time than in earlier studies, and were required to perform more complex tasks involving query construction and reading, evaluation and labeling of returned documents.

The studies mentioned above on the significance of time as an indicator of user interests appear to suffer from two factors: first, the studies considered average reading times, and not individual user reading times. Average times may be indicative of group preferences, but they do not say much about individual user preferences, since different people read at different rates, and even the same person may read at different rates at different times, or depending on the nature of the material they are reading. Second, the studies did not account adequately for varying document lengths. One can expect longer documents to take longer to read than shorter ones. It would be interesting to find the relationship between individual users' reading times—perhaps normalized by document length—and their interests in the documents.

Maximal Forward Reference (MFR). Another implicit indicator of user interests is maximal forward reference (Chen, Park, & Yu, 1996), in the domain of web navigation. This approach is based on tracking users' forward and backward references as they navigate a website. A forward reference is a page not in the set of pages already visited by the user in the current browsing session. A backward reference is a page that is already in the set of pages for the current session. A maximal forward reference is the last forward reference page requested by a user before backtracking occurs, where the user requests a page previously viewed during that session, and is considered a page of interest to the user.

Edit Wear and Read Wear: Hill, Hollan, Dave, and McCandless (1992) used the idea of physical wear that an object experiences with use, to describe *edit wear* and *read wear* as useful relevance clues left behind by documents that experience extensive use by authors (edit wear) or users (read wear).

Examination, Retention and Reference: These interests indicators were examined by Nichols (1998) and Oard and Kim (1998).

Examination comprises the following user behaviors: selection of documents for further examination; reading time; edit wear to describe for example, interests in particular sections of a document based on amount of scrolling; repetition to characterize repeated actions; purchase, an important interest indicator in situations where the user is willing to pay for an item.

Retention groups behaviors that suggest some interest to make use of an object in the future. This category comprises the following user behaviors: save (creating a bookmark, creating a symbolic link in a file system or saving an object with or without annotation, perhaps in a special folder as opposed to the default folder); printing; deleting (e.g., deleting an email to indicate lack of interest in that email).

Reference includes user behaviors that create implicit or explicit links between information objects, for example, forwarding email, replying or posting follow-up messages in a discussion group, following hypertext links or cited documents, cutting and pasting.

2.2.3.5. Sources of Data Used to Infer User Profile Implicitly

There are three primary sources of data used to infer web user profiles implicitly: the client computer, the user's navigation context, and user access logs.

Collecting Profiling Data from the Client Computer. User profiling data can be collected from the client computer through the use of cookies[6] (Pirolli, Pitkow, & Rao, 1996) or remote agents[7]. The main problem with these approaches is that users may to block them from their machines.

Collecting Profiling Data from User Navigation Context. The context of the current user's navigation can also be used as the source of user profiling data. This approach is useful in systems that do not keep a persistent record of user navigation, for example, when browsing an encyclopedia on CD. Hirashima, Hachiya, Kashihara, and Toyoda (1997) for example, used *context-sensitive* filtering, which involved combining the index terms and recency of pages visited as the basis for the user profile.

Collecting Profiling Data from User Access Logs. Implicit user modeling data can also be obtained from user access logs stored in server logs. This is useful for example, in systems that model users based on

[6] A cookie is a server-generated identifier stored on the client machine, used to overcome the problem with statelessness of the HTTP protocol by tracking user interaction with the server.
[7] Just like a cookie, a remote agent is a program that resides on the client machine. But unlike a cookie, the remote agent provides usage information on user interaction not only with the server that generated it, but with other servers as well.

web server access logs. Most server logs amongst other things, maintain information on the pages accessed, the time each page was first accessed, and the IP address of the computer from which the request was made to the web server. Using server log data has the advantage of a plentiful supply of modeling data.

2.2.3.6. Representation of User Profiles

The data corresponding to user profiles must have appropriate representation for use by the system. In knowledge-based recommender systems, the representation makes use of any knowledge structure that can be used to draw inferences from the user's interaction with the system. Entrée (Burke, 2001, 2002) for example, uses techniques from case-based reasoning[8].

In behavior-based systems, the representation could be in the form of stereotypes or overlays. Stereotyping (Brusilovsky, 1996; Rich, 1979) is a simple but effective user modeling approach in systems that need to adapt to different classes of users. A stereotype comprises a set of related attribute-value pairs. The user is assigned to a stereotypical group based for example, on demographic information. The same attribute-value pair defines each stereotype, and a user assigned to a stereotype inherits all the attribute-value pairs of that stereotype, and in the case of a recommender system, receives the same recommendations. One attraction in the use of stereotypes in modeling the user stems from the fact that some of the modeling data may not be readily available and so the user is classified by grouping him with other people who have other similar attributes to his. The main disadvantage of the use of stereotypes is that of large granularity: minor but possibly important differences are not taken into consideration.

Overlays (Brusilovsky, 1996, 2000) are more powerful and flexible than stereotypes in representing user interests, and are more widely used as a result. The model estimates how much the user is interested in an item, by using a set of attribute-value pairs. Each value in the pair is an estimation of how strongly the attribute or concept can be associated to the user, with the values taking discrete or probabilistic values. Examples of the use of the overlay model include: the use of vectors of term weights in content-based systems to represent a user profile, and the use of item rating vectors to represent a user profile in collaborative filtering systems. Stereotypes and overlays can be combined to give better results, for example, using a stereotype to define the initial user model, and an overlay to update the model.

[8] A case-based reasoning (CBR) system uses the recall of examples (cases) as the fundamental problem-solving process. It comprises a number of *knowledge containers*: the *case-base*, the *vocabulary* used to describe cases, the *similarity measure* used to compare cases, and the knowledge needed to *transform* recalled solutions.

2.2.4. Associated Technologies in Recommender System Design

Recommender system design usually involves one or more of a large number of technologies. This section discusses the following commonly used ones: data mining, machine learning, IR, use of graphs, and clustering.

2.2.4.1. Data Mining Tools

Data mining, or *knowledge discovery in databases (KDD)*, is the automated or convenient extraction of patterns representing knowledge implicitly stored in large databases, data warehouses, and other massive information repositories (Han & Kamber, 2001). Application areas of data miming include customer profiling, fraud detection, evaluation of retail promotions by commercial enterprises, credit risk analysis, market segmentation[9], and now increasingly, web mining.

The application of data mining techniques to discover knowledge from web data has recently become popular partly because of the large amounts of such data available. Cooley, Mobasher, and Srivastava (1997) defined *web mining* as the "discovery and analysis of useful information from the World Wide Web." They distinguished between *web content mining*, which involves automatic on-line search for information, and *web usage mining*, which involves discovery of user access patterns.

Web content mining approaches are either agent-based or make use of the database approach. Agent-based approaches include: the use of intelligent search agents that conduct searches based on user profiles; IF and categorization agents that retrieve, filter and categorize documents; and the use of personalized web agents that learn user preferences and return personalized information to them based on their individual preferences. The database approach attempts to organize semi-structured web data into a more structured collection, suitable for database-like querying.

Web usage mining relies on the navigation patterns of past users of a website, stored in web server log files. These server logs are an invaluable source of useful information, as they can be analyzed for purposes ranging from improving marketing strategies to better structuring of websites and hyperlink recommendation. Web usage mining involves one or both of the following two steps: discovery of significant access patterns (or user navigation behaviors), and analysis (e.g. interpretation or visualization) of the discovered access patterns. The mining is based on the following premise: if it is known that several users have visited the same set of pages, then it can be said with some certainty that the pages are related. In recommender systems, this knowledge is used as the basis for making recommendations to an active user.

One common goal of data mining is the discovery of association rules, or interesting correlations among a set of data items (also known as *transactions*) in a database. An illustrative example of the use of

[9] Market segmentation is the selection of groups of people who will be most receptive to a product. Segmentation methods make use of demographic variables (age, sex, race, income, occupation, education, household status, and geographic location), psychographic variables (life-style, activities, interests, and opinions) and product use patterns.

association rules is in market basket analysis[10], where the mining algorithm seeks to find groups of items that customers tend to buy together, perhaps to help with better planning of the shop display.

2.2.4.2. Machine Learning of User Profile

Machine learning investigates the mechanisms by which knowledge is acquired through experience. In recommender systems design, machine-learning techniques are commonly used to learn a model that is used to profile users as they use the system. Three commonly used machine learning approaches include k-nearest neighbor algorithms, use of neural networks, and the use of Bayesian networks.

K Nearest Neighbors (KNN) Algorithms. A KNN algorithm (Sarwar et al., 2000b, 2001) is a memory-based classification algorithm which, given an object *O*, searches the space of classified documents for the class of objects which has the greatest number of members that are similar to object *O*, and assigns object *O* to that class. The algorithm uses three components: a positive integer *k*, a similarity function *D*, and a training set of *n* correctly labeled feature vectors $Sn = \{(x_1,C_1),(x_2,C_2),\ldots(x_n,C_n)\}$ where x_i is vector *i* and C_i is the class to which x_i belongs. Given an unclassified feature vector x, the algorithm identifies the subset of *k* vectors that are closest to x (the k nearest neighbors) based on the similarity metric *D*. Then, of the *k* nearest neighboring vectors, the class that occurs with greatest frequency is determined and vector x assigned to that class.

The first stage in KNN classification is identification of the *k* nearest neighbors, or *neighborhood formation*. The goal is to find for vector x, an ordered list of *k* vectors $X = \{X_1, X_2,..., X_k\}$ such that $x \notin X$ and *sim*(x,X_1) is highest, *sim*(x,X_2) is next highest, etc., where *sim* is the similarity measure. Two commonly used neighborhood formation schemes are *center-based* and *aggregate* (Sarwar, Karypis, Konstan, & Riedl, 2000a). The center-based scheme forms a neighborhood by selecting the *k* vectors that are most similar to x. The aggregate neighborhood scheme uses the current members of the neighborhood to determine the future members of the neighborhood. Using this scheme, the neighborhood is formed by first selecting the closest vector to x, and then using the following steps to select the other *k-1* neighbors.

- Let there be *j* (*j*<*k*) vectors currently in the neighborhood x
- Compute the centroid of x: $\vec{C} \quad \frac{1}{j} \quad {}_{\vec{v} \quad x} \vec{v}$
- Select vector w $\notin$ x as j+1th neighbor if w is closest to $\vec{C}$
- Repeat until $|\mathrm{x}| = k$

KNN classifiers are *instance-based* or *lazy* learners because they store all the training data in memory, and only proceed to build a classifier at problem solving time when a new unlabelled document arrives to be

[10] Market Basket Analysis is an algorithm that analyzes customer buying habits by finding associations between the different items that are most frequently purchased together (Han & Kamber, 2001). It takes its name from the idea of a person in a supermarket putting the items they intend to buy in a shopping cart (i.e., a "market basket").

classified. This is in contrast to *eager* learners like neural networks, which construct a generalization of the model before new samples arrive for classification. There is a tradeoff between training and classification time in lazy and eager learners: lazy learners have faster training times but slower classification times, and the converse is true for eager learners (McKenna & Smyth, 2000). If the number of potential nearest neighbors in a KNN algorithm is large, performance may significantly degrade since matching to the nearest neighbor is done online.

Neural Networks. Neural networks (Swingler, 1996) are statistical models of real world systems, built by tuning a set of weights. The model maps inputs from the world to corresponding outputs, with the weights of the network defining a model of the system.

A neural network learns a model of the system through *training by example*: input and output pairs are fed into the system and the weights adjusted to minimize the error (i.e., difference between the system output, and the known, correct output). After the network is trained, its weights are frozen, and a test set of unseen input/output pairs fed into the network. If the error between the network's outputs and the correct outputs in the test set is acceptably small, the network is assumed to generalize well to all other unseen inputs.

The most common class of neural networks is the multi-layer perceptron (MLP). It comprises a set of small computational units arranged in three or more layers: an input layer that takes input from the world, and passes them through interconnecting weights to one or more hidden layers which in turn, pass the resultant signals through connecting weights to the output layer, where the results of the model are read. The hidden layer units extract useful features from the input data, and form an internal representation of the corresponding outputs. An example of an MLP is shown in Figure 4.

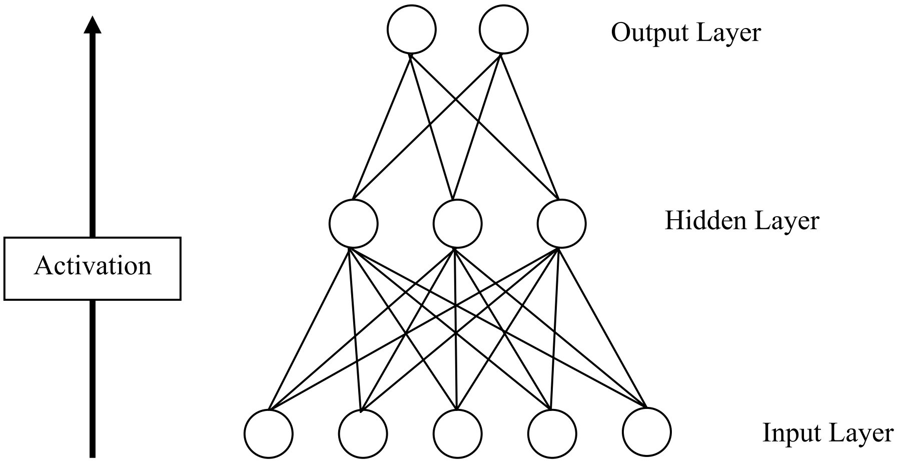


Figure 4. A fully connected acyclic neural network.

An MLP is fully connected if each unit of a layer is connected to each of the units in the next layer, and to no other units. An MLP could also be acyclic, if there is no feedback from the output of any unit to the input of a unit at a lower layer, or cyclic if there is at least one such feedback. Figure 4 for example, is a fully connected, acyclic neural network, with one hidden layer.

The value, or activation, of each unit flows to the units above it through the connecting weights. The activation received by a unit receiving activation from one or more lower level units is determined by computing the sum of products of the lower level activations and the connecting weights and passing the results through a *squashing* function that bounds the activation to a given range, usually –1 to +1.

The most widely used weight adjustment (or learning) procedure for MLPs is a gradient descent approach called *back propagation of error* (Rumelhart, Hinton, & Williams, 1986). The procedure involves determining the error at each output unit, assigning blame for the error proportionately to the connecting weights and their corresponding lower level activations, and adjusting the weights such that the error would be smaller the next time the same activations are provided to the output unit. The blame (or error) attributed to each hidden layer unit is similarly back propagated to the connecting lower level units until the input layer is reached.

Bayesian Classifiers. Bayesian classifiers apply Bayes Theorem to classify items. Bayes Theorem is itself based on the intuition that "a hypothesis is confirmed by any body of data that its truth renders probable" (James, 2003). Two commonly used Bayesian classifiers are the naïve Bayesian classifier and the Bayesian network.

The naïve Bayesian classifier is a supervised learning algorithm that stores a single probabilistic summary for each class. The algorithm makes the assumption that the attributes of a class are independent and that the classes are mutually exclusive and exhaustive. In spite of these naïve assumptions, naïve Bayesian classifiers have been found to be useful in practice. For example Miyahara and Pazzani (2000) developed a naïve Bayesian classifier that performed collaborative filtering on users' preferences for movies.

Belief (Bayesian) networks (Clemen, 1996) store knowledge in a directed acyclic graph in which nodes correspond to attributes and links indicate dependencies between attributes. The Lumiere Project (Horvitz, Breese, Heckerman, Hovel, & Rommelse, 1998) which formed the basis of the *Office Assistant* in the Microsoft Office suite of application programs for example, used Bayesian user models to infer a user's needs by considering the user's background, actions, and queries.

2.2.4.3. Information Retrieval Tools

IR techniques, especially the vector space *Term Frequency Inverse Document Frequency* (TFIDF) model (Salton, 1989; Salton & McGill, 1983; Salton & Wong, 1975), are heavily used in recommender system design. The TFIDF model represents a document as an *N*-dimensional vector, where *N* is the number of distinct terms in the collection of documents being considered. Each term is represented in the document space as a weight based on heuristics that are described shortly. The similarity between two documents, where a document could be one of the documents in the collection, a user query, a user profile vector, or any *N*-

dimensional representation of the terms of the document collection, is determined by any of several similarity measures.

The vector space model comprises three stages. The first stage involves document indexing. Indexing involves extracting the features of the document, typically words. Because of difficulties involved in working in a high-dimensional space and because many words in a document do not sufficiently describe the document content, dimensionality of the document representation is reduced by passing the terms of the document collection as parameters of a stop-list algorithm to remove the less descriptive terms. Dimensionality is also reduced by eliminating very high- or very low-frequency terms, which do not carry much information.

The second stage assigns weights to the index terms to enhance retrieval of relevant documents when a query is issued, or to improve filtering results based on the user profile under consideration. Assignment of term weights assumes that the importance of a term within a document is proportional to the frequency of its occurrence within the document and inversely proportional to its frequency across the document collection, hence the name TFIDF. The intuition used in assigning the term weights is as follows: the frequency of occurrence of terms in a particular document is proportional to its importance in the document; but given a collection of documents, terms that occur in a large number of different documents do not help much in discriminating between different documents, hence the inverse proportion. In order to avoid giving preference to longer documents in which terms are likely going to appear more frequently, document lengths are normalized. The weight given to any term is typically proportional to the product of the term frequency, inverse document frequency and the length normalization factor.

The last stage of the model involves determining the similarity between the document and query vectors in the case of IR or between the document and user profile vectors in the case of a filtering system. The similarity is determined from associative coefficients based on the inner product of the document vector and query or user profile vector. The most commonly used measure of similarity between two documents is the cosine similarity coefficient[11], which measures the cosine of the angle between the two document vectors. Other measures of similarity include the Pearson correlation index, Jaccard coefficients, and Dice coefficients. Typically, documents returned by the model are presented as a list, ranked in order of similarity (i.e. relevance) to the user query or profile.

One major criticism of the vector space model is its bag-of-words approach, which refers to the fact that the model simply considers the occurrences of words in the document with no regard to the semantics or syntax associated with them. Nevertheless, the approach is widely used and has proven successful in IR and in machine learning algorithms despite this apparent limitation.

[11] The cosine similarity measure was used throughout this work to determine the similarity between any two vectors. The steps involved in computing cosine similarity are shown in Appendix A.

2.2.4.4. Graph Techniques

Pirolli, Pitkow, and Rao (1996) presented a number of uses to which graph structures can be put to identify usable structures in the web, which recommender system design can benefit from. Graph structures used to represent web pages comprise nodes representing web pages, and arcs representing the strength of association between the web pages.

One of the most commonly used graph structures for web pages stores the hyperlink topology of a website or collection of sites. In this representation, directed arcs are used to connect one web page to another if a hyperlink links the former to the latter. This representation could be useful in recommender systems if there is the need to know if a given page *p* is reachable from a set of pages *S*, in which case, the task would be to determine if an edge in the graph exists between any of the nodes present in *S* and Node *p*.

Another use of graph structures is to represent the similarities between web pages, or user behaviors. In such a representation, arcs between nodes are labeled with the similarities between web pages. This representation could be useful in determining which pages to cluster together, and so treat them similarly.

Yet another graph representation could be used to show the frequencies of user accesses from one node to another. In this representation, directed arcs between nodes are labeled with the access frequencies between them.

2.2.4.5. Clustering

Clustering is an unsupervised learning task: items are assigned to classes (clusters) without any prior knowledge of what class the items belong to (Han & Kamber, 2001). Cluster analysis aims at finding structure in data by grouping together (or clustering), data items that are highly similar to each other, and very dissimilar from data items in other clusters (Aldenderfer & Blashfield, 1984; Han & Kamber, 2001; Kaufman & Rousseeuw, 1990). In recommender systems, clustering may be used to classify similar data together so that recommendations could be made only from clusters that have been determined to be interesting. This could lead to the design of systems that scale well due to considerable savings on the time used to generate recommendations, which could be considerable if the complete document space had to be searched for relevant documents.

Five broad categories of clustering methods are in use. These are partitioning (or iterative) methods, hierarchical methods (agglomerative and divisive hierarchical clustering), density-based methods, grid-based methods, and model-based methods.

Partitioning methods result in the creation of k ($k < n$) clusters out of n objects. These methods involve the creation of an initial cluster, and then iteratively, up to k partitions are created by moving objects from one group to another more suitable group. The exhaustive enumeration of all possible partitions can be very expensive, and in practice, heuristic approaches are used to determine partitions. These heuristic approaches include *k-means* and *k-medoids* algorithms. *K*-means represents a cluster by the mean value, and attempts to move objects to the clusters that would result in the minimum intercluster similarities. One weakness with k-means is it sensitivity to outliers. The *k*-medoid algorithm attempts to reduce this sensitivity by taking the medoid, or most centrally located object to represent a cluster.

One problem with the partitioning approach is the difficulty of correctly estimating the number of partitions to create. Hierarchical clustering methods create a hierarchical decomposition of a data set. Two approaches are commonly followed: agglomerative and divisive. The agglomerative approach starts with the assumption that every object belongs to a separate cluster, and iteratively merges objects or groups that are similar to each other until all the groups are merged or a terminating condition met. The divisive approach assumes that there is initially only one cluster, and the algorithm iteratively splits clusters into smaller units until every object is in a separate cluster or until a terminating condition is reached. Examples of hierarchical clustering algorithms are AGNES (AGglomerative NESting) and DIANA (DIvisive ANAlysis). The main disadvantage of hierarchical clustering is that once an object is merged into a cluster, it cannot be reassigned to another cluster.

Density-based clustering algorithms assume that clusters are dense regions of data—represented by high density data points—separated by noisy regions—represented by low density data points. These algorithms can discover clusters of arbitrary shape and are commonly used in spatial databases.

Grid-based clustering approaches quantize the data space into a number of cells on which clustering is done. The main advantage of these approaches is speed since clustering time depends on the number of cells, and not the number of data points present. Typically, statistical information on the data in each cell are precomputed and stored for later use.

Model-based clustering methods attempt to fit the data into some mathematical model. Two approaches are commonly used: a statistical approach that uses probability theory to determine what cluster an object belongs to, and a neural network approach. The neural network approach uses an exemplar (a prototype vector) to represent each cluster, and uses some similarity measure to place a new object to the cluster that is most similar to it.

2.2.5. Recommender Systems in the Context of Other Related Technologies

Recommender Systems in the Context of Information Retrieval and Information Filtering. At an abstract level, both IR and IF have the goal of retrieving information that is relevant to the user's information needs while minimizing the amount of irrelevant information retrieved (Folz & Dumais, 1992; Hanani et al., 2001; Malone et al., 1987; Oard, 1997). But how they go about fulfilling this goal differs. Belkin and Croft (1992) emphasized the similarities between IF and IR which led them to conclude that the two technologies "are in fact two sides of the same coin." Belkin and Croft (1992), Folz and Dumais (1992), and Hanani et al. (2001) also pointed out some differences between IR and IF. These similarities and differences, and how they relate to recommender systems, are presented in Figure 5.

As can be seen from the figure, recommender systems are quite similar to IF systems. Recommender systems can be viewed as a kind of filtering system in which the user profile is very dynamic (and not static), as in the typical IF system. Recommender systems may be considered as active – using autonomous agents to continuously traverse the information space and collect items to recommend or classify (Sarwar et al., 1998), or interface agents that watch and learn from users – and eventually automate some of the tasks that users do routinely (Maes, 1994). Recommender systems may also be considered passive, such as those used to make recommendations on email messages. These recommenders accept all incoming messages and do not actively search for information for the user.

Based on similarities between IR and IF

Similarity	Information Retrieval	Information Filtering	Recommender Systems
Need to satisfy an information need	IR users specify their information needs in the form of queries	User information needs are specified in the form of user profiles	User information needs are specified in the form of user profiles
Use of text[12] surrogates to represent information sources	Representation involves creation of indexes from text	Representation is the text produced and distributed to users with matching profiles	Representation is the text produced and distributed to users whose current profile state matches the text
Matching of information need and text surrogate	Typically involves comparison of user query with the text surrogate representing the document	Filtering process involves comparison of the user profile with the text surrogate representing the document	Available documents are filtered based on the current state of the changing user profile
Use and Evaluation of retrieved text	Evaluation may lead to query modification in a process called relevance feedback	User evaluation involves updating his/her profile. If system maintains profile, then the system will need to be able to learn from the user behavior, how to update the profile.	System learns from user behavior and updates profile (and thus, recommendations) accordingly

Based on differences between IR and IF

Difference	Information Retrieval	Information Filtering	Recommender Systems
Frequency of use	Designed for ad-hoc use by persons with a one-time search goal	Designed for repeated use by long term users with long term information needs	Depends on system. Some systems (e.g., news recommenders) maintain long term user profile; others (e.g., web browsing), do not.
Process of determination of user information	User actively specifies information need in a query	User not actively involved in retrieving relevant information once profile is defined	User not actively involved in retrieving relevant information once profile is defined
Goal	Selection of relevant data from databases	System concerned with removal of irrelevant data from incoming data streams, or collection and distribution of relevant data from specified sources.	System concerned with selection and presentation of most relevant information to user as he interacts with the system
Data source	Relatively static[13] (e.g. CDROM)	Dynamic	Dynamic

[12] Information sources are normally text or multimedia. Research on retrieval and filtering on non-text documents is still in its infancy and throughout this report, information sources and information surrogates are assumed to be purely text.

[13] With the explosion of WWW usage and content, this assumption is no longer true; search engines need to, and do constantly monitor available content on the web.

Types of users	Not known by the system	Known to the system by their profiles.	Known to the system by their profiles.
Privacy issues	Not of much concern; users not profiled	Important, as users are profiled	Important, as users are profiled

Figure 5. Comparison of recommender systems and IR & IF, based on comparison of IR & IF by Belkin and Croft (1992), Folz and Dumais (1992), and Hanani et al. (2001).

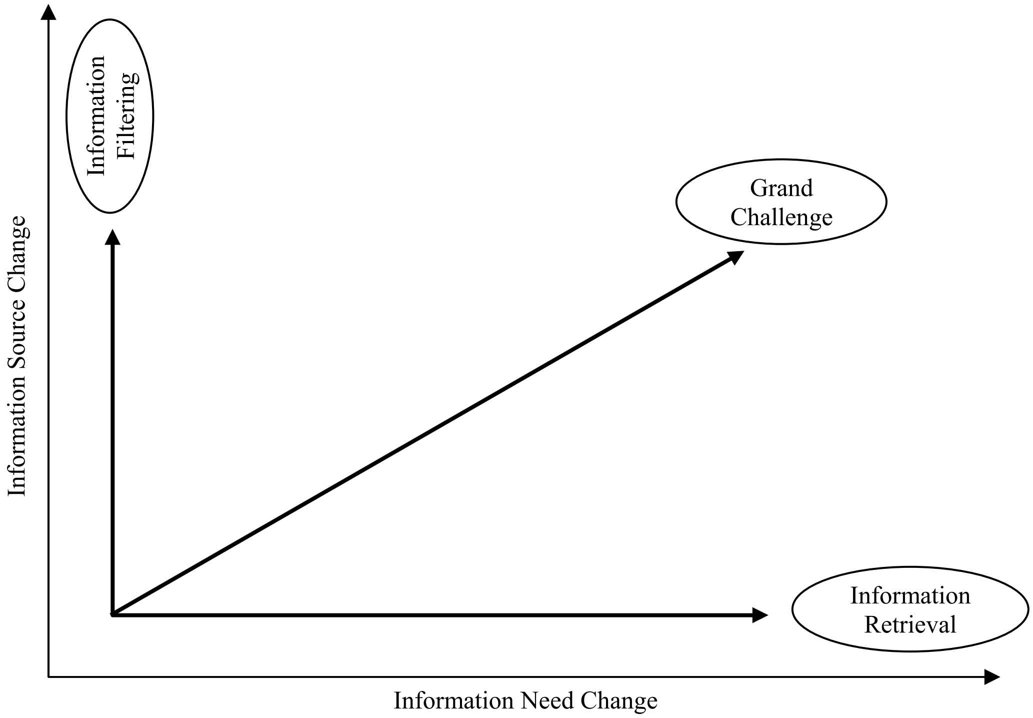


Figure *6*. Information Seeking Space – adapted from Oard (1997).

Oard (1997) presented a useful way to visualize the areas in information space that are occupied by most IF and IR systems (see Figure 6). As can be seen from the diagram, IF systems typically have dynamic information sources while the user needs are fairly constant; on the other hand, IR systems traditionally deal with stable information sources, but changing user needs. Figure 6 also shows that both IR (stable information sources but changing user needs) and IF (stable user needs but changing information sources) are specializations of a more difficult problem of dealing with both changing information sources and changing user needs – the *grand challenge*. The recommendation problem may be considered to be somewhat more challenging than the search or filtering problem alone. As in IF, the information source may change continuously, but the system must also monitor changes in the user profile. Similarly, as in IR systems with changing user queries, the user profile changes in recommender systems; unlike in IR systems in which the user specifies his changing interest in a new query however, recommender systems are responsible for keeping track of these changes.

Recommender System in the Context of Adaptive Hypermedia. AH research aims to develop hypermedia systems that personalize their behavior to goals, tasks, interests, and other features of individual users and groups of users (Brusilovsky & Maybury, 2002). They help users navigate through hyperspace by providing adaptation in three main areas: adaptive content selection, adaptive navigation support and adaptive presentation.

Adaptive content selection may be used for example, in a search system where the role of adaptation is to select and prioritize the most relevant candidate items to present (Brusilovsky & Maybury, 2002).

Adaptive navigation support may involve direct guidance (e.g. the use of *Next* and *Previous* buttons); adaptive link sorting, presenting the most relevant links first; adaptive link hiding (hiding, disabling and/or removal of links); adaptive link annotation, where particularly interested links are specially annotated; adaptive link generation, where relevant links that were not preauthored on a page are generated in the course of browsing the page; and map adaptation, i.e., adapting the hypermedia map that is presented to the user (Brusilovsky, 1996, 2001; Brusilovsky & Maybury, 2002).

Adaptive presentation deals with adaptation of the content that is presented. This may involve adaptive text presentation, adaptive multimedia presentation and adaptation of modality, which is useful if there is need to choose between different types of presentation forms—i.e., audio, video or text (Brusilovsky, 1996, 2001; Brusilovsky & Maybury, 2002).

As in recommendation systems, AH systems make use of a user profile to determine what kind of adaptation to perform. The main difference between the AH and other information seeking processes like search, recommendation and filtering systems, is that AH systems typically use the output of these other systems as the basis of adaptation. For example, a hyperlink recommendation system generates a set of hyperlinks to present to a user; adapting the hypermedia would typically involve determination of how the generated links are presented to different users. Or the output of a search engine may be adapted by sorting in order of relevance before display.

2.2.6. Recommender Systems Application Areas

This section presents a sample of past and present recommender systems for web navigation, and a cross-section of other domains in which recommender systems are used.

2.2.6.1. Web Navigation

Web navigation recommendation takes the form of recommended hyperlinks for users to follow. Depending on the system, recommendations are either site-specific, or unrestricted to the hypermedia, e.g., for the whole web. In the former case, although coverage is much narrower, the system designer has much greater control of possible system inputs, and there is a better chance of constructing an accurate recommendation model, while the converse situation holds in the latter case.

Siteseer. Siteseer (Rucker & Polanco, 1997) is a web page collaborative filtering recommendation system that uses users' bookmarks and the organization of bookmarks within folders as the basis of recommending relevant pages. Siteseer capitalizes on the fact that bookmarks, especially those placed in a folder are the result of an intentional act hence are less noisy as a source of inference of user intention. Disadvantages of the system included the following: several pages important to the user are never bookmarked; some bookmarks are only used to indicate transient interest in a page; bookmarks cannot be used to indicate a lack of interest.

The user profile comprises a combination of folders and the various URLs stored in them. Siteuser uses a nearest neighbor algorithm to find a pool of reviewers qualified as trusted recommenders. The process involves determining the similarity between different users' bookmark folders by determining the percentage of common URLs in them. The system's recommendations comprise pages that the active user's virtual neighbors have bookmarked, with preference given to pages in folders with the greatest overlap and also pages occurring in multiple folders.

Fab. Fab (Balabanovic & Shoham, 1997b), is an adaptive multi-agent hybrid recommender system that uses content-based and collaborative filtering techniques to recommend web pages to users. User profiles are based on analyzing the contents of documents that the user has rated on a 7-point scale in the past. The collaborative component of the hybrid is achieved by comparing different user profiles to determine neighboring users for collaborative recommendation. Items are recommended if they are rated highly both by the active user and by other similar users.

The content-based part of the system makes use of 3 components: collection agents that find web pages for a specific topic for a group of users, selection agents that find pages from the pool of collected pages for a specific user, and a central router. For each user, each collection agent maintains a keyword-based profile for a particular topic, with the various topics dynamically generated as clusters based on the characteristics of all the users of the system. Collection agents that consistently find documents that users rate poorly, or that users do not bother to rate, are purged, while more successful collection agents are duplicated to replace them.

Selection agents maintain keyword-based profiles of individual users. They receive (via the router) matching pages from the collection agents and do some processing, like discarding already seen pages. Explicit user ratings of received documents are used to update the selection agents' profile.

Letizia. Letizia (Lieberman, 1995; Lieberman, Fry, & Weitzman, 2001) is a reconnaissance agent[14] that tracks a user's browsing behavior while at the same time searches the links from the current web page, to determine potentials pages to recommend when the user requests recommendations from the system. Recommendations are generated following content analysis of candidate (i.e., potential) pages.

[14] A reconnaissance agent searches new *territory* (web pages in this case) before the user ever gets there.

Simple heuristics are used to determine *interestingness* of a page and to guide the search. Letizia's search is performed breadth-first for hyperlinks that the system may recommend while visiting the current page. Breadth-first search compensates for the fact that users tend to browse depth-first often missing useful neighboring links on previous pages, which Letizia can recommend any time the user asks for recommendations. The system considers factors like book marking of a page, following a link from the current page, and repeatedly returning to the same page, as indicators of the interestingness of the page. Rapidly leaving a page is assumed to indicate lack of interest in that page while following a link that is farther away in a left–right, top–bottom direction is considered lack of interest in the preceding links.

Letizia maintains a set of keywords from the pages the user has visited with weights corresponding to the relevance of the words to the user's profile. By comparing this profile with vector representations of pages that the agent searches, a similarity measure is determined, which acts as a measure of the usefulness of the page to the user if he requests recommendations.

IndexFinder. IndexFinder, (Perkowitz & Etzioni, 2000) is the recommendation engine of an AH system that automatically generates index pages to help a webmaster transform a site by adding the generated pages to the index page if they are judged to be appropriate, or to reject them if he does not deem them necessary.

IndexFinder makes the assumption that an index page should have links to related pages, that is, pages that cover a particular topic. Generating an index page involves determination of related but unlinked pages, where two pages are considered linked if there is a link from one to the other or if there exists a page that links to both of them. The aim is to generate for each index page, all the links that are related to a particular topic, but no link that is unrelated to the topic.

Adaptation is not based on a single user's profile, but rather on a profile based on the consistent navigation pattern of a large group of users. IndexFinder uses a statistical cluster mining technique that depends on the co-occurrence of web page requests during user visits, to deduce candidate page clusters from web server logs. Each candidate cluster is fed to a concept-learning algorithm that produces a concept description for the cluster, comprising a conjunction of attribute-value pairs, e.g. *instrument type = drums AND file type = audio*, for a cluster that should contain music files involving drums. Pages in a candidate cluster that do not fit the concept description are then discarded. The resulting cluster is then presented to the webmaster as an HTML page of sorted links, which he can provide a header for and a link from an appropriate page in the site, or which he can reject.

ProfBuilder. Profile Builder (ProfBuilder) (Ahmad, 1999) is a hybrid recommender system that combines collaborative and content-based filtering and the use of autonomous agents to recommend interesting pages within a website to users. Profbuilder maintains user profiles based on the vector space model, using keywords as tokens. Each time the user visits a new page, his profile is updated thus: $Q_{j\_new} = Q_j + t_{ij} * D_i$, where Q_i is the vector space representation of the user profile, t_{ij} is the weighting factor for document *I* and user profile *j*, and D_i is the vector space representation of the current page.

Learning the user profile involves inferring the value of t_{ij}. The method is based on determining the conditional probability that a user will visit page *B*, given that she visited another page, *A*, based on evidence from access log data. Using classical information theory, the value of information of a page is then quantified as a decreasing function (or entropy) of this conditional probability. Hence t_{ij} is precomputed for various *i* and *j* values from data stored in user log data.

ProfBuilder performs collaborative filtering based on finite context modeling on the active user's navigation pattern. The procedure depends on the assumption that the current page being browsed depends on a context consisting of a finite set of *m* preceding pages. Collaborative filtering is performed by comparing the user's current navigational context to navigation patterns of previous users and using the conditional probabilities of visiting a page given the recent navigation history, as a measure of the user's ratings for pages.

Content-based filtering is performed by correlating the vector representations of contents of pages in the website and the user profile.

Profbuilder does not merge recommendations from the two filtering techniques into a single list; instead, two separate lists are generated, one for each recommendation technique.

Calvin. Calvin (T. Bauer & Leake, 2002) is a multi-agent content-based recommender system that recommends web pages used in similar prior browsing contexts. It aims particularly to provide useful recommendations in situations that answer questions similar to the following: is a user who is browsing a page about a topic (say intelligent agents) in a company's website, interested in agents technology or is he trying to learn more about the company for some other reason like applying for a job? Calvin takes into consideration other pages that the user has browsed to determine what his real interests are before making recommendations.

Calvin uses *WordSieve*, a real time competitive learning text analysis algorithm to learn keywords that partition the user's access patterns into different contexts and uses these keywords to characterize his current browsing context. The user profile comprises these access patterns, which WordSieve uses to index individual documents and identify similar contexts in future. The keywords are arranged in a hierarchy with the first layer comprising words that occur frequently in the current document stream, the second layer comprising words that occur frequently over broader periods of time and the third layer comprising words that occasionally cease occurring, making them candidates for partitioning sets of related documents.

Webwatcher. Webwatcher (Armstrong, Freitag, Joachims, & Mitchell, 1995) is an information search agent that interactively assists and advises users as they search for information by following hyperlinks. As a user browses the web, Webwatcher builds a log of his navigation history, makes recommendations on possible interesting links found on the currently accessed page, and learns from both the links that the user follows and those that he does not.

The user starts by filling a form with keywords that describe the information he is looking for, e.g., author name, article title, subject area, author's institution, etc., for a user searching for an article. As the user starts browsing, Webwatcher learns which links on the current page to recommend by computing a link utility function for every link on the current page. The utility function estimates the probability of the usefulness of a link on the current page, for given a search goal and user.

The profile of the generic user is represented as a Boolean vector of about 530 features (words). The first 200 features are the most informative that are determined from the words of the hyperlinks recorded in the currently logged navigation history. The next 200 features are the most informative that are extracted from the words of the sentences of the current page that contain hyperlinks. The next 100 features are allocated to indicate selected keywords that appear in the page heading. The last set of words (about 30), indicate words entered by the user as he specified his search goals. Armed with this information, Webwatcher uses a machine learning algorithm to determine which hyperlinks on the current page if any, to recommend.

Syskill & Webert. Syskill & Webert (Pazzani & Billsus, 1997; Pazzani, Muramatsu, & Billsus, 1996) is a content-based recommender that makes use of an intelligent agent to learn users' long term information needs, and uses that knowledge to recommend interesting web pages to them.

A user profile is determined for each user for each topic. A topic is created and named by the user, as well as the URL of an index page (a page with several links to other pages that deal with the topic). User profiles comprise the most informative words on the topic. These words are learned by determining the expected information gain that the presence or absence of the word would have on the classification of the page. These profiles may be optionally initialized by words provided by the user.

The user may rate each page as hot (two thumbs up), mild (one thumb up and one down), or cold (two thumbs down), vis-a-vis the current topic, although internally, the agent combines the mild and cold pages to a single category, cold.

Syskill & Webert may then be requested to make recommendations from the links present on the current page. Alternatively, the agent may be requested to construct a query comprising the most frequently occurring words and the words that most sufficiently discriminate between topics, to submit to the Lycos search engine to find interesting pages, and evaluate the returned pages to determine their interestingness.

The user profile is learned by constructing a binary word vector indicating the absence or presence of the selected informative words of a topic, and combining this vector with the user's rating for that page in a naive Bayesian classifier that is used to update the profile.

2.2.6.2. Other Domains

E-commerce

The use of recommenders in E-commerce is usually to entice users to buy items from an E-commerce site, and to come back in future. Amazon.com's book recommender system (Schafer, Konstan, & Riedl, 1999) for example comprises several features which provide a wide variety of functions, including information of buying habits of similar customers, an email notification system to inform users of new items, customer ratings of books that are used in collaborative recommendation, and a facility for customers to make informed buying decisions based on comments of other users of the system.

Movies

MovieLens. MovieLens (2002) is one of the most successful music recommendation system currently in use. It is a collaborative filtering system with a user base of several thousands, and with millions of recommendations, which can be downloaded by researchers. PolyLens (O'Connor et al., 2001) is an extension of MovieLens, designed to present group recommendations.

Restaurant Recommendation

Entrée. Entree (Burke, 2002) is an interactive knowledge-based recommender system that uses case-based reasoning to select and rank restaurants. The user submits either a known restaurant or a set of criteria, and the system returns one main recommendation plus a short list of similar (neighboring) restaurants based on the following attributes, each of which the user can critique to change recommendations: cuisine, price, quality and atmosphere. Entrée is stateless and does not store a user profile; its response is determined by the chosen example restaurant and the user critique.

Recommendation is based on knowledge-based similarity retrieval. The technique involves *similarity-finding*, which returns a sorted list of similar restaurants (based on attributes like *niceness*) from the system database, and *critique-based navigation*, which filters out similar restaurants that do not meet the user's critique.

The similarity measure for each attribute is an integer based on simple heuristics. For example, a more expensive restaurant is generally considered worse than a less expensive one, and a more highly rated restaurant better than a lower rated one. The most complex similarity measure used is on the cuisine attribute. This metric is based on a semantic network of 150 cuisines, with the similarity values considered inversely proportional to the distance separating them in the network.

EntréeC is a knowledge-based/collaborative hybrid recommender system derived from Entrée. The collaborative component gets its ratings implicitly by watching the user's navigational actions and deciphering ratings (positive or negative based on those actions).

Three approaches to finding similar users based on implicit indicators are used. First, a coarse-grained approach that uses four interest indicators is used to weight the user interest in each restaurant in a scale of –1 to +1: entry point, i.e,. starting point for browsing is given a rating of 1; exit point, i.e., user either found the restaurant he wanted or simply gave up, rated at 0.8; browsing, i.e. a negative interest indicator since user is navigating away from the current restaurant, rated at –0.5; and critiquing a restaurant, rated at –1. Different users' profiles are then correlated to find similarities between them.

The next approach is finer-grained and aims to avoid losing semantic details that the coarser-grained approach encounters. The approach depends on a binary vector comprising various levels of each of the attributes specified for each restaurant. The difficulty posed by this approach is that the user profiles are now represented by a very large multidimensional, sparse matrix, the size of which increases rapidly as more restaurants are added to the system.

Finally, EntreeC makes use of a *heuristic-similarity* approach which takes account of the semantics of the ratings themselves. For example, a livelier restaurant is considered to be not only different but opposite to a quieter one.

On-line Recruitment

CASPER. CASPER (Rafter, Bradley, & Smyth, 2000), is a recommender system that uses implicit user profiling data as collaborative filtering data, and uses that knowledge to make job recommendations to users. CASPER is integrated in an Irish job search website, JobFinder (http://www.jobfinder.ie/). The user profile is constructed by mining JobFinder's server log. The server log holds data on user ID, jobs visited, and type and time of access to the job. For each user, a profile is constructed which comprises jobs visited, and for each job, the action taken (e.g. read, apply on line, email to self), number of revisits to the same job and read time. To avoid the sparsity problem of collaborative filtering systems and the scaling problems related to the KNN approach, a pre-processing clustering stage is used to identify virtual communities of similar profiles. The online recommendation stage then makes use of items rated highly for users in the same virtual community as the active user.

TV Viewing

We preview two systems that recommend TV programs to users based on their viewing habits: ClixSmart and TV Scout.

ClixSmart (Smyth & Cotter, 2000) is a content-based/collaborative filtering hybrid recommender system that recommends TV programs to users. The collaborative filtering component of the system makes program recommendations based on the similarities between a user's ratings for programs to those of other users with similar ratings, while the content-based component supplements recommendations by comparing the textual attributes (program name, channel, viewing time, textual description) of recommended programs to the same attributes of other programs, and selecting those that are similar as additional recommendations.

TV Scout (Baudisch & Brueckner, 2002) also makes TV program recommendations using a collaborative filtering/content-based hybrid. Content filtering is prominent earlier on, when the user receives recommendations based on the result of queries he selects from a menu of system-stored queries. The user may then select or ignore the programs recommended based on the queries he selects. The collaborative filtering component interprets each of the user's selections as a vote for that program, and generates ratings in relation to the query or queries that generated it, before applying collaborative filtering techniques to compute a recommendations list.

Research Paper Recommendation

Quickstep. Quickstep (E. S. Middleton et al., May 2002; S. Middleton et al., 2001) is a content-based/collaborative filtering hybrid recommender system that combines knowledge-based and behavioral approaches to user profile construction, to recommend research papers to users. Domain knowledge of the topic (or class) of computer science-related papers is obtained by classifying an initial set of papers into an ontology. Future papers are classified into one of these classes using a nearest neighbor algorithm.

The user profile is a vector of weights that indicate the user's interests in particular topics. The weight for each topic is updated in two ways. The first approach is by adding a numeric score to the weight of the class of the currently selected paper, based on what action the user takes on a paper that belongs to that topic: 1 for simple browsing; 2 if the user follows a recommended page; 10 for a positive rating; –10 for a negative rating. The second method used to update the profile is by using a decay function that decreases topic weights by the inverse of the number of days since the user last browsed a page in that class.

2.3. Improving Performance of Recommender Systems

One of the main problems faced by recommender systems is that of poor scaling, resulting from the need to handle very high dimensional data. In collaborative filtering systems, nearest-neighbor algorithms do not scale well for large and sparse databases; in content-based filtering systems making use of the vector space model, it is difficult to handle high dimensional vectors because a typical object will only exhibit a small subset of the features, causing document vectors to be virtually indistinguishable from each other. Performance may be improved by a number of preprocessing techniques, including the following: dimensionality reduction; use of filtering agents to populate a sparse collaborative filtering user-item matrix; clustering of documents to limit the amount of data that needs to be consulted at recommendation time to matching clusters; and mining of the available data for information that could be useful at recommendation time. The rest of this section discusses dimensionality reduction and use of filtering agents; data mining and clustering techniques were presented in Sections 2.2.4.1 and 2.2.4.5 respectively.

2.3.1.1. Dimensionality Reduction

Learning in a high-dimensional space – or the "curse of dimensionality" (Lang, 1995) is a problem because many more examples are needed to determine which dimensions are important in the document space. The aim of dimensionality reduction of a CF system ratings matrix, or the term frequency matrix of the vector space model, is to create a search space that is small enough for real time computations to be completed in reasonable time.

One commonly used technique for dimensionality reduction is *singular value decomposition* (SVD). The technique involves reducing a high-dimensional matrix into the product of three matrices $T_0 S_0 D_0^{'}$, such that the columns of T_0 and D_0 are orthonormal to each other and S_0 is diagonal. Latent Semantic Indexing (LSI) (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990) is an example of a SVD dimensionality reduction technique. LSI makes the assumption that there is some underlying latent semantic structure in data that is partially obscured in the bag-of-words representation of documents. Using SVD, a *semantic space*[15] is constructed from the term-document matrix, placing closely related terms and documents together, while ignoring less important term-document associations. Position in the semantic space acts as semantic indexing, and determination of relevant documents involves identifying a point in the space that is represented by the terms in the query in IR, or the user profile in personalization systems.

Principal Component analysis is another dimensionality reduction technique. It is a statistical procedure that can identify the directions (principal components) along which data components are correlated. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. Directions with little variability can then be ignored at the cost of a small error.

Another dimensionality reduction approach involves the use of auto-encoding neural networks, which attempt to reduce the document representation space while at the same time retain the information contained in the original document. The autoencoder network is strictly layered, comprising *N* input units, *K<N* hidden units and *N* output units whose values should match the values of the input unit. For linear autoencoders, the reduction in dimensions takes place at the hidden layer.

2.3.1.2. Use of Filtering Agents

Filtering agents can be used to evaluate and rate each item in a collaborative filtering system, resulting in a dense rating. This alleviates the sparsity and early rater problems in collaborative filtering systems. In the Usenet domain, Sarwar et al. (1998) used automated rating robots called *filterbots*, to evaluate new documents as soon as they were published and rate them using a number of algorithms. The agents *behaved* like ordinary users of the system, but because they provided ratings for articles as soon as the articles were published, real users had sufficient data that the system could compare with theirs. If a filterbot's ratings matched the ratings of a real user, then recommendations originating from the agent could be made for the user.

The system by Sarwar et al. (1998) made use of three filterbots to verify the hypothesis that filterbots can improve user utility: a *Spellcheckerbot*, an *IncludedMsgBot* and a *Lengthbot*. The spellcheckerbot rated

[15] The meaning of semantic here is limited to the proximity of similar terms and documents which is assumed to entail some latent information; it does not refer to the meaning of the terms themselves.

articles based on the proportion of spelling errors present in the article text, assigning higher ratings to articles with fewer errors. The IncludedMsgBot rated each article based on the percentage of text quoted from other articles, using the intuition that users dislike long messages with little new content, and so are not happy if a discussion thread is made unnecessarily long by including previously seen message in the text. The lengthbot rated articles based on the hypothesis that Usenet readers valued brevity, and so rated shorter articles higher than longer ones.

The authors found that in four of the five newsgroups they considered, there was at least one filterbot that improved user utility for two metrics: coverage and accuracy[16].

2.4. Recommender Systems and Privacy

When users visit websites, especially e-commerce sites, they leave behind a trail of information (Schafer, Konstan, & Riedl, 2001), including the following:

- Preference information explicitly provided in the form of product ratings, comments, or attributes defining their interests
- Preference information implicitly provided in the form of products and information viewed, time spent viewing, kinds of searches performed, etc.
- If purchases are made, transactional information like type of payment, credit card account, shipping address, products purchased, etc.
- Explicitly provided identification information like name, email address, and telephone number
- Implicitly provided information like machine IP address, or domain from which the user is browsing

In addition, companies can acquire a lot more information about users from other companies.

These factors make recommenders a “double-edged sword”: while they promise benefits like greater convenience, they also have the potential of invading personal privacy. Personalizing a website requires that information about users be known, and the deeper this knowledge, the more personalized the experience can be. But used wrongly, for example, if personal information is made available to third parties, this gives the third parties the opportunity to invade user privacy in ways ranging from inundating the user with unwanted mail, to unwanted phone calls, or even unauthorized financial transactions. So there is a clear need to balance

[16] Coverage measured the percentage of items for which a recommendation system can provide recommendations. Accuracy measured how effectively recommendations helped users select high-quality items.

users' desire for personalized service with their concerns about privacy (Riedl, 2001). This section discusses some of the privacy issues related to recommender and other personalization systems.

Some websites attempt to achieve the balance between their need for information and the user's need for privacy by making use of privacy policies. A site's privacy policy usually includes one or more of the following:

- A promise not to sell personal information to third parties without the user's prior permission
- A promise not to use the customer's email address for advertising and not to call their phone numbers for purposes other than in connection with the current transaction.

But privacy policies alone do not provide sufficient protection or reassurance for the user. Reasons for this include the following: the very confusing nature of the legal language used; the fact that websites reserve the right to change their privacy policies without notice, or to use user information for unspecified "business interests" of the company; and the danger of private information falling into unanticipated hands as businesses consolidate.

Riedl (2001) and Schafer et al. (2001) argued for privacy assurance to be based on the Platform for Privacy Preferences (P3P) initiative of the WWW consortium (W3C). P3P is designed to be integrated into browsers to let users easily manage relationships with multiple websites and identify what types of information they are willing to share with each. Using the P3P protocol, a site creates a machine readable version of its privacy policy in a format that computers can understand and negotiate about privacy. A user agent (usually the browser) is entrusted with the responsibility to negotiate the user's privacy demands with the sites he visits. For P3P to work, a third party would be required to enforce these privacy policies by auditing businesses to ensure that they are following their stated policies. The down side to such centralized control however, is that it would serve as a convenient target for hackers to try to access information about millions of people from a single site.

Singh (2001) discussed the privacy-related conflict that results from the advancement and merging of two technologies that have somewhat conflicting philosophies: communications and Internet technologies. The success of the Internet has been largely due to its openness – users need not divulge information they consider privileged, or even operate under their true identities unless it suits their purposes to do so. Telecommunication companies on the other hand, operate a closed service, in which individual telephones are attributable to individual users; these services lack the anonymity of the Internet. With the development of Wireless Applications Protocol (WAP) based services, telephone operators can now also determine the exact geographic location of their customers. And there is pressure from governments for telecommunication companies to make geographical locations of the origin of all nature of telephone calls available to emergency services in case of need. But the same equipment that makes geographic information readily available can also be used to determine customer location when there is no emergency. There is thus the possibility for invasion of privacy if this information is abused. This leaves not only customers vulnerable, but also the telephone companies, which may face expensive litigations, even if customer privacy was violated by a third party.

Singh (2002) proposed the use of a subscriber agent and one or more service agents to deal with these problems that result from this potential compromise in user privacy. The subscriber agent represents the user's interests, while the service agents negotiate with the subscriber agent regarding what information and authorizations can be made available to third parties, in return for a given quality of service.

Claypool, Brown, Le, and Waseda (2001) highlighted privacy concerns resulting from the implicit acquisition of user profiles through devices like web usage mining, java scripts embedded in web pages, cookies, and Active X controls. The benefit of unobtrusively collecting user profiling data comes at a cost to personal privacy, because implicit interest indicators also lend themselves to potential privacy abuses. Because it is possible to predict a user's interest based on his navigational behavior, this information could be misused if it falls into the wrong hands, for example, a rival CEO.

Faber (2001a; 2001b) highlighted two privacy-related issues: first, the conflict between the user's need for privacy on the one hand, and the government's for more information on citizens, and second, the need to address security (and privacy) problems inherent in the design of communications infrastructure.

On the first problem, citizens demand that their governments provide them with safety without compromising their privacy, but the government argues that they cannot guarantee safety if government is not allowed to learn even more about the citizenry. These conflicting demands became very apparent following the September 11 2001 terrorist attacks on the US.

The second problem highlighted by Faber deals with the lack of security on the Internet, primarily because of the way the technology developed. The Internet started as a university research project with the aim of linking universities together, and priority was given to a working, and not a secure system. As the Internet grew, there was little time and energy available to address the security problems, and as a result, most security-related problems are today simply patched, not solved. Faber pointed out an opportunity with the arrival of fast end-to-end optical communications links, and argued that information security and network resource security should be a fundamental requirement in the design of these new networks.

The recommendation technique used may also be a source of concern for privacy. Ramakrishnan, Keller, Mirza, Grama, and Karypis (2001) identified a privacy risk faced by *straddlers*, or users of collaborative filtering systems, with eclectic tastes. The risk stems from the ability of these recommenders to make serendipitous or cross-genre recommendations. If there is a small number of users with distinct interests and a person receives one or more recommendations relating to these interests, it could be possible in certain circumstances with some probing, to correctly guess the identities of other users whose profiles are responsible for these recommendation(s).

3. PROBLEM STATEMENT

Catledge and Pitkow (1995) identified three browsing strategies for a website user: (1) search browsing: directed search, where the goal is known; (2) general purpose browsing: consulting sources that have a high likelihood of items of interest; and (3) serendipitous browsing, which is purely random. The goal of personalization is to provide users with what they want without requiring them to ask for it explicitly (Mulvenna et al., 2000). In the context of web navigation, personalization can be achieved by conveniently presenting to a user, hyperlinks that are *relevant* to that user's current navigation goals. These relevant hyperlinks are recommendations that are computed by the system on behalf of the current user, and should be tailored to that user's specific needs. While hyperlink recommendations are meaningless for the serendipitous user, users engaged in search and general purpose browsing can reap significant benefits from such recommendations.

Recommended hyperlinks can be limited to the pages available in a particular website or collection of websites (e.g., Profbuilder (Ahmad, 1999); Calvin (T. Bauer & Leake, 2002)), or to the whole web (e.g., Fab (Balabanovic & Shoham, 1997b); Webwatcher (Armstrong et al., 1995); Syskill & Webert (Pazzani & Billsus, 1997; Pazzani et al., 1996)). This work considers the case of generating hyperlink recommendations for a user browsing a single website. We limit our recommendations to hard-coded hyperlinks. We recognize the fact that an increasing number of web pages that users receive in their browsers are generated dynamically, but we do not attempt to personalize the delivery of these pages. We expect nevertheless, that personalization based on static hyperlinks will improve the overall navigation experience.

The problem of deciding on relevant hyperlinks to recommend is a difficult one largely because relevance is a subjective measure. IR researchers have studied a number of approaches to try to make the documents returned in a search task more relevant. Relevance feedback (Rocchio, 1971) for example, is based on the assumption that although users cannot adequately specify their needs in a query, they can recognize relevant information when it is retrieved. Such user feedback can then be used to reformulate the query and retrieve even more relevant documents.

In a navigation task a user follows hyperlinks the labeling of which suggest to the user that he is moving to pages that will meet his current information needs. Hyperlink labels are conceived by the website designer to serve as local cues that users process in making judgments on which hyperlinks to follow. In foraging theory (Card et al., 2001; Larson & Czerwinski, 1998; Pirolli, 1997; Pirolli & Fu, 2003), these cues are referred to as information scent[17]. But information scent is a subjective measure. In the domain of web navigation, the difficulty is that there is usually a conflict between the website designer's view of how the site should be navigated, and the actual navigation paths that users follow. Just as the user in a search task is expected to correctly recognize relevant feedback from search engines, it is hoped that in a navigation task, the user will be able to recognize relevant hyperlink recommendations. If that is the case, the user should follow these recommended links if they take him to pages that are of interest to him, faster than if he relied solely on hard-coded hyperlinks, some of which may be several mouse clicks away.

This work is based on the notion that *navigation behaviors* of past users can be used to model the navigation behaviors of current users. We equate a navigation behavior to any set of HTML pages that users of a website visit together frequently, irrespective of the sequence in which the pages are visited[18]. A navigation behavior may comprise a user session, or only part of a session. We seek to discover frequent, past user navigation behaviors which could then be compared to those of current users, recommending to current users the same web pages that past users with similar navigation behaviors are believed to have been interested in. The fundamental assumption is that users interested in the same web pages exhibit similar navigation behaviors as they browse a website, and so if a statistically significant number of past users are known to have had a similar navigation behavior, then the web pages that those past users were interested in can be recommended to a current user with similar navigation behavior. We used user access log data collected by web servers as users visit a website to model past user navigation behaviors.

The problems tackled in this research are the following:

- Analysis of some existing approaches and technologies currently used in the design of hyperlink recommender systems based on past users' navigation behaviors
- Proposals on improvements to existing approaches
- Proposals of some new approaches
- Implementation and evaluation of the improvements and new approaches mentioned above

The website of the School of Information Sciences, University of Pittsburgh (SIS), henceforth referred to as "the website", was used to test the ideas presented in this work.

[17] The concept of information scent characterizes how users—like organisms that use scent to determine where go next—evaluate the utility of hypermedia actions to lead them to their target information. The greater the scent, the more likely is the user to access the target information.

Pirolli (1997) notion of information scent is similar to Furnas' (1997) use of the term "residue" to refer to the cues left behind in the text or graphic used to represent the distal object that lies behind a hyperlink. Card et al. (2001) defined information scent as a user's "(imperfect) perception of the value, cost, or access path of information sources obtained from proximal cues, such as WWW links."

[18] Researchers like Zimdars, Chickering, and Meek (2001) have argued that more meaningful decisions on data such as web user navigation patterns and TV viewing histories can be made if the temporal nature of the data is taken into account. In this research however, navigation behaviors were handled using the simpler bag-of-words approach.

The rest of this section presents the following: difficulties involved in modeling past user behaviors using user access logs and heuristics used by past researchers to overcome the difficulties, some of which are adopted in this work (Section 3.1); and the specific research objectives of this study, detailing similarities to, and differences from similar works in the past (Section 3.2).

3.1. Using Web Server Access Log Data to Model Past User Navigation Behaviors: Difficulties and Useful Heuristics

Past user access logs are a rich source of data that can be used to learn which web pages to recommend to a user browsing a website. By applying a web usage mining procedure on user access log data, general rules describing the relationship between past users' navigation behaviors and the corresponding pages that they found interesting can be discovered. A recommender system could then be designed that recommends these interesting pages to active users with similar navigation behaviors. Following work on web page classification (Cooley et al., 1997; Cooley, Mobasher, & Srivastava, 1999; Mobasher, Cooley, & Srivastava, 2000; Mobasher, Dai, Luo, & Nakagawa, 2001; Pirolli et al., 1996; Pitkow, 1997), we refer to the interesting pages as *content pages*, and the pages that are traversed before getting to content pages as *navigation pages*.

There are several levels of difficulty involved in the discovery of navigation and content pages from web server user access logs. The first is determining the *identities*[19] of users of the website. The second is determining all the web pages (or navigation paths) accessed by each user, each time they browse the website, and the third, determining the various navigation behaviors (referred to as *transactions* elsewhere in this report) exhibited by the user for a given session. These difficulties, and heuristics used to address them, are presented below as the *user identification problem*, *navigation path identification problem*, and *transactions identification problem*.

User Identification Problem. User identification is difficult because most web servers do not require user authentication. Even for sites that require user authentication, as Pitkow (1997) points out, it cannot be certain that the information provided by users is accurate and reliable, because users are very concerned about privacy, and to protect themselves, may register multiple times, or provide misleading information.

One way to try to identify users is by using the IP addresses of their machines as a first step. Problems in using IP addresses include errors introduced if the client machine is shared by more than one user, which makes the task of identifying different users difficult; difficulty identifying users whose machines are located behind a proxy server or firewall, which results in a single IP address (the IP address of the proxy server or firewall) reaching the web server.

[19] Identity here refers to the attribution of web page accesses to the correct user without necessarily knowing the true identity of that user.

Navigation Path Identification Problem. Even if all page accesses recorded in user access logs could be correctly attributed to the right users, it is still hard to know all of the pages that a user visited during a session because of problems relating to caching between users' browsers and the web server. Pitkow (1997) and Pirolli et al. (1996) discussed difficulties in tracking a user's navigation path.

The first problem is related to the way the HTTP protocol (Fielding et al., 1999) works. A single HTML page requested from a web server may contain other URLs, like image, sound and video files, and applets, embedded in it. In order to retrieve these additional files, the HTTP protocol makes a separate connection to the web server to retrieve each of them, and as a result, a separate entry is made into the server log files for each of these requests. The problem this introduces to navigation path identification takes several forms. First, the number of additional pages retrieved differs, and depends on the number of embedded URLs that the web page designer placed in the requested page. This has an impact on path identification first as it results in varying path lengths for the same number of pages requested. Second, the order in which the embedded pages are retrieved is not the same, even if the only difference between two web pages is the order in which the embedded documents are placed in them. This would have an impact on path identification if the sequence of pages accessed is an important consideration. Third, some of the embedded pages could be pages that the user may sometimes explicitly request, and the web server logs would have no way of distinguishing a page that is explicitly requested from one that just happens to be embedded in a requested page.

Another problem that hinders accurate tracking of a user's navigation behavior is the failure of some user requests to be recorded in web server logs. This problem results from the presence of caches at proxy servers and at the user's browser. Normally, caches improve client-server interaction by making available, cached pages, instead of requiring a new connection to the web server for every page request.

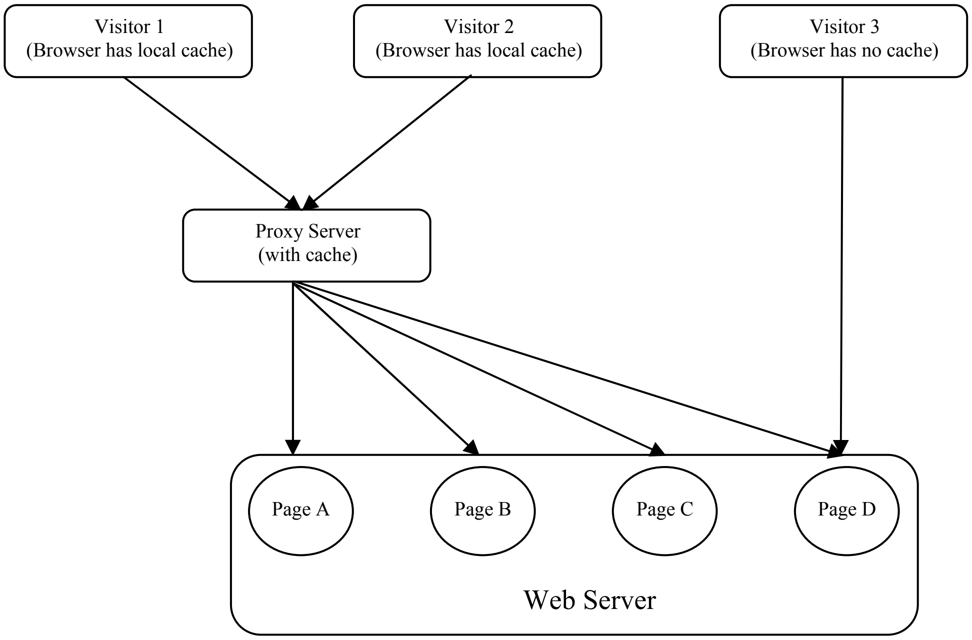


Figure *7*. Illustrating the effect of caches on data recorded in web server logs. Source: Pitkow (1997).

Caches introduce problems though, as illustrated in Figure 7, which shows 3 visitors accessing a website with two of their machines behind a proxy server, and the third connected directly to the web server. Both the proxy server and the computers used by *Visitor 1* and *Visitor 2* have caches, while *Visitor 3*'s computer has no local cache. If *Visitor 1* requests *Page A*, followed by *Page B*, and then uses his browser's *Back* button to return to *Page A*, only two page accesses, one for *Page A*, and the other for *Page B*, will be recorded in the user access log file because the second access to *Page A* will be provided by the local browser cache. On the other hand, an identical sequence of page accesses by *Visitor 3* will result in three page accesses recorded in the server logs, two for *Page A* and one for *Page B* because the absence of a local cache means that every page request from *Visitor 3* will reach the web server.

Another problem caused by browser caching can be appreciated if one considers a simple scenario where pages *A*, *B*, *C*, and *D* of Figure 7 are linked as shown in Figure 8. If *Visitor 1*, who started off by accessing pages *A* and *B*, followed by the browser's *Back* button, also visits *Page D*, then if referrer logs are not kept by the web server, it would not be possible to tell from the server logs if he reached *Page D* via the hyperlink on *Page A*, or that on *Page B*.

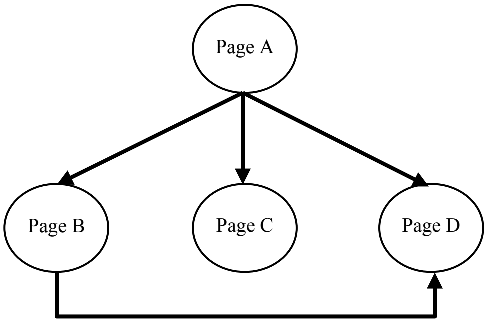


Figure *8*. Illustrating difficulties in identifying the correct sequence of linked page accesses from web server logs. Source: Pitkow (1997).

Yet another problem caused by caching at the proxy server can be illustrated by examining what happens when 2 requests are made for the same page, the first from *Visitor 1* and the second from *Visitor 2*. This sequence of requests results in only one entry in the user access logs because the first request from *Visitor 1* places the page in the proxy server cache, and the second request from *Visitor 2* results in a page fetch from the proxy server cache, and never reaches the web server.

Another difficulty in determining navigation paths from access log data is determining when one session ends, and when another begins. This would not be much of a problem if the time separating sessions with the same IP address were large; this, in general, is not the case.

A commonly used heuristic to reduce the problem of correctly identifying separate navigation paths is to define session boundaries (Chen et al., 1996; Cooley et al., 1997, 1999; Schechter, Krishnan, & Smith, 1998), by using a timeout mechanism to determine when a current session must end. Thirty minutes is commonly used to define the maximum session window for a user. Hence, for the same IP address, the first web page access beyond the 30 minute window is assumed to denote the start of a new session.

Pirolli et al. (1996), Pitkow (1997), and Cooley et al. (1997) presented a number of heuristics for identifying unique users and user sessions from web server logs.

- For the same IP address, if a request is known to come from a different browser or operating system, then it can be assumed that there is a different user with the same IP address (Cooley et al., 1997).
- For a given IP address if a page is requested that is not directly reachable by hyperlink from one of the pages already viewed by the user, it may be assumed that the request is coming from a different user (Cooley et al., 1997; Pirolli et al., 1996; Pitkow, 1997). It should be noted that this assumption breaks down if the user arrived the page by directly typing the URL instead of following a hyperlink. But because typing the URL is not a typical browsing strategy, this category of errors is expected to be few, and should not severely affect the user or session identification process. Pirolli et al. (1996) and Pitkow (1997) suggested that a least recently used

(LRU) policy be used to attribute a page to a user if there is ambiguity as to which of several users with the same IP address, made the request. Knowledge of whether a page is reachable from another page can be obtained from the topology of the website, which shows which and how pages are linked to each other.

Transactions Identification Problem. A user session comprises one or more transactions. Identification of transactions—or user behaviors—from session data is complicated by the fact that web pages cannot be classified strictly as navigation or content pages. Some pages that serve as navigation pages for some navigation behaviors can serve as content pages for other behaviors, and vice versa.

Chen et al. (1996) introduced the notion of *maximal forward reference* (also see Section 2.2.3.4, Page 17) as a useful indicator of content pages from a stream of navigation and content pages. The idea is to track users' *forward* and *backward* references to determine maximal forward reference (or content) pages, each of which denotes the end of a transaction.

Consider Figure 9 for example, which shows a graph representation of a small portion of a website comprising nine web pages *A – I* as nodes, and edges connecting nodes that are linked to each other; we ignore the direction of the links for now. If a user navigates this portion of the website by visiting pages in the order *A-B-C-D-C-F-G-F-I-B*, then the maximal forward references are *D*, *G* and *I*. The next forward reference following a maximal forward reference signals the start of a new navigation path, while the maximal forward reference page at the end of a navigation path is assumed to denote a page of interest for that navigation behavior.

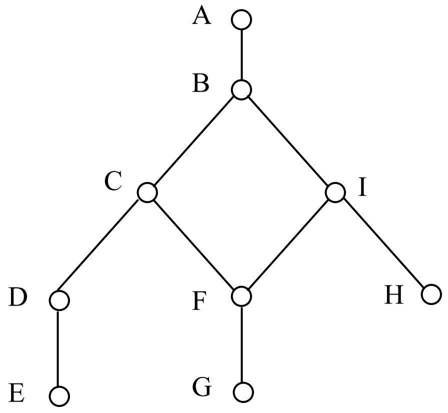


Figure 9. Portion of a sample web site used to illustrate maximal forward reference.

3.2. Research Objectives

Consider a user navigating a website, part of which link structure is shown in the directed graph in Figure 10, where the nodes are web pages and each directed edge denotes a hyperlink from one page to another. The numbers indicate the relative number of visits that each node is known to witness over time. Now consider the problem of recommending to a user, what pages to visit in the site, assuming the user starts at node A. A naïve approach to making recommendations to this user would be to recommend *Page B* when the user is at *Node A*, and assuming the user navigates to *Page B*, to recommend *Page D*, because these recommended pages are the most likely to be accessed for the general user.

There are three problems with this approach to making hyperlink recommendations. First, the recommendation process is reduced to annotating the currently accessed web page with the most likely hyperlinks that the general user follows, which may be very different from the pages that are of interest to the user. Second, relying solely on the statistics of likely page accesses from the current node does not capture the history of the user's navigation up to the current node, a source of useful information on which page the user is likely to visit next. Finally, with this approach, only hyperlinks that are present on the current web page can be recommended; potentially useful web pages that are not linked to from the current page cannot be recommended.

It is desirable to design a hyperlink recommender system (1) that is capable of recommending not only pages that are directly linked to from the current page, but other pages as well, if such pages are deemed interesting; (2) to use the history of the user's navigation up to the current node as a guide to what pages to recommend next. Assume for example that it is known that the user has visited pages *A* and *B*, and we know from past experience that 80% of users who visited pages *A* and *B* proceeded to *Page C*. For such users, *Page C*, and not *D*, is the preferred page to recommend.

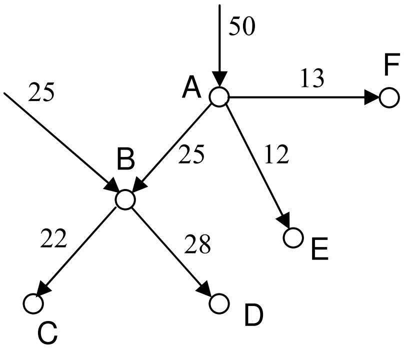


Figure 10. Sample website showing relative frequencies of node accesses.

A flowchart of key steps involved in the design of a hyperlink recommender system based on past user navigation behaviors is shown in Figure 11. As can be seen from the figure, the design involves offline and online processes. This is important because recommendation is a live process, and the more of the computations that are done offline, the better the recommendation latency, and the better the chance of user acceptance of the system. The goal of the system is to use past user navigation behaviors to predict the web pages current users with similar navigation behaviors would be interested in, and then compute a recommendation score for each of those pages. The clustering stage is not mandatory, but if present, could be used to influence the construction of a prediction model, or for augmenting recommendations by suggesting other pages from the same cluster as recommended pages.

This research is limited to the off-line section of the figure. The specific problems tackled in this work are introduced below.

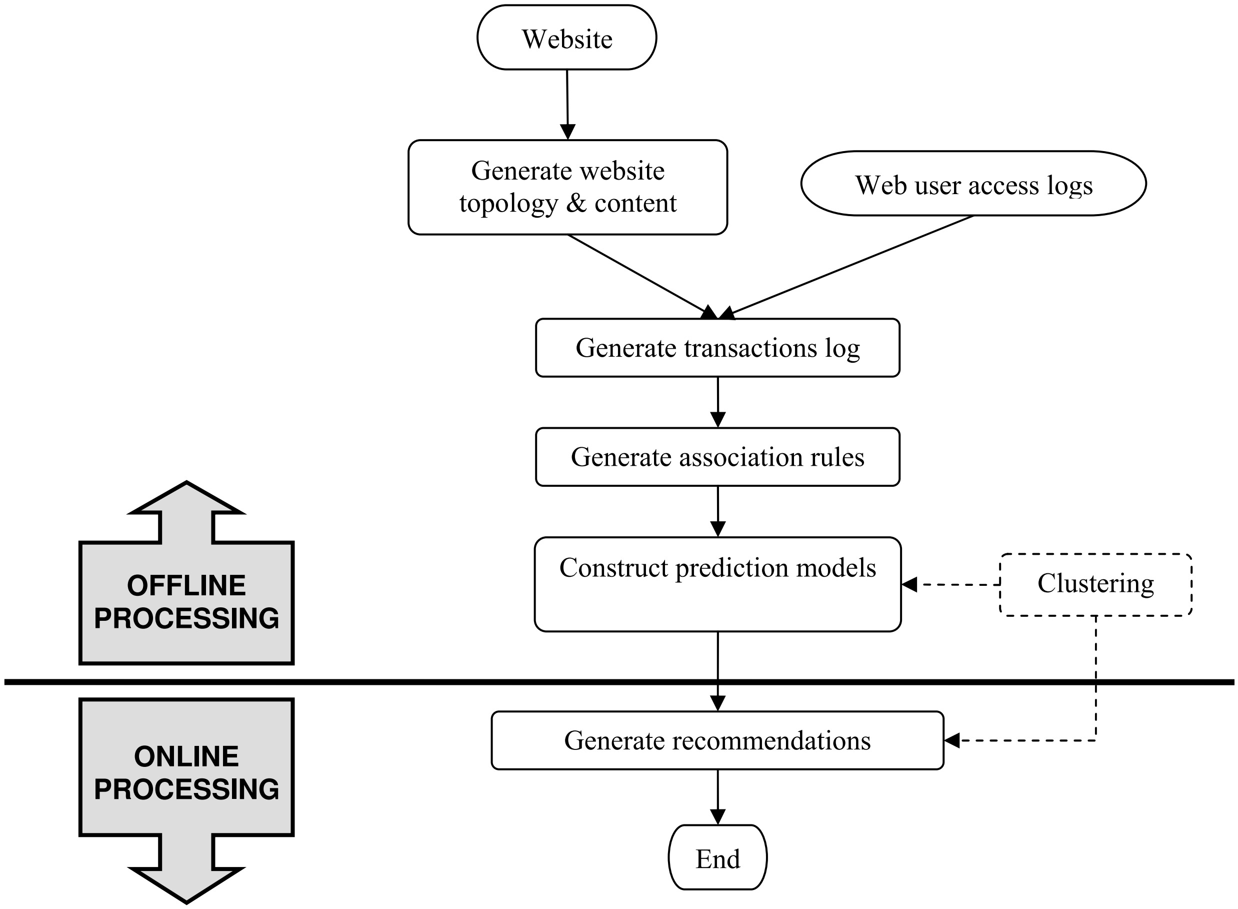

Website
Generate website topology & content
Web user access logs
Generate transactions log
Generate association rules
Construct prediction models
Clustering
OFFLINE PROCESSING
ONLINE PROCESSING
Generate recommendations
End

Figure 11. Steps involved in the design of a hyperlink recommender system based on past user navigation behaviors.

Generation of Transactions from Web Server Logs. The transactions log is the collection of past user behaviors (i.e., series navigation pages, followed by a content page) extracted from web server logs. We applied the heuristics presented in Section 3.1 to extract past user behaviors from web server logs as a starting point, and then investigated an additional, new heuristic, *Page Rank by Inverse Links-to-Word count ratio –*

PR × ILW (Nkweteyim, 2005) to use to classify web pages as content or navigation, and thus determine additional navigation/content page boundaries for very long transactions. We then ran a small user study to compare web page classification by humans to classification using the new approach.

Generation of Association Rules. The association rules show the relationships between past user navigation behaviors and the pages that they were interested in. We investigated the apriori algorithm that has been influential in the mining of association rules, and demonstrated that a modification of the classical apriori algorithm that aims to reduce the space and time complexities of the algorithm is possible. We then ran simulations to compare performance of the classical apriori algorithm to the new one.

Construction of Prediction Models. As mentioned earlier, the basis of hyperlink recommendation in this study was the similarity between the current user's navigation behavior to those of past users of the website. The aim of each prediction model was to predict what pages the current user would be interested in, given his current navigation behavior determined by the last *N* pages visited by the user. Once these predictions were made, the recommendation engine could then compute which of them to recommend, and the order in which the recommendations should be presented. Two approaches to the construction of prediction model were studied, both based on collaborative filtering. The two models used KNN algorithms to compute predictions but differed in the way the neighborhood was computed: the first model used the center-based and the second, the aggregate-based neighborhood formation schemes (see Section 2.2.4.2).

Clustering. Commercially available clustering algorithms do not perform well when there are more than about 1000 objects to cluster, or when the dimensionality of the objects to cluster is very high. In this research, we implemented the CLARANS (Clustering Large Applications based on RANdomized Search) clustering algorithm to cluster the web pages of the web site based on similarity of the page contents. We also performed a simulation study to compare CLARANS to two other clustering algorithms: CLARA (Clustering LARge Applications) and PAM (PArtitioning about the Medoids), on which CLARANS is based. Such a study is not new, but traditionally, CLARANS has been applied to spatial, low-dimensional objects; this study involved the clustering of high-dimensional data objects.

The rest of this section shows in more detail, the objectives that this study sought to accomplish.

3.2.1. Transactions Generation Objectives

As mentioned above, we propose PR × ILW as a new metric to help in classifying web pages as content or navigation. The motivation for this new metric was the following: Yan et al. (1996) in their work, which tracked user web navigation behavior, found that 50% of users accessed 2 pages or more, 20% accessed 5 pages or more, and under 10% accessed 10 pages or more, to get to a page of interest. These results suggest that most users follow between two and five hyperlinks before reaching a page of interest. Our initial results of discovering navigation and content pages based on the MFR heuristic showed that there were several cases where the transactions were still very long, some longer than 150 pages. There were two possible reasons for such situations: they could represent users whose browsing did not involve a return to a page seen earlier; or the transactions may have resulted from failure of a session or transaction identification heuristic to correctly predict a past user's navigation behavior. We speculated that these long transactions contained one or more *hidden* content (and thus recommendable) pages, which MFR alone was not able to detect. The challenge therefore, was to find a heuristic that could be used to make plausible guesses on what these hidden content pages were. Our approach combined the ratio hyperlink count:term count of web pages, and the topology of the web site to compute a number that determined whether a page within a very long transaction should be classified as content or navigation, and hence determine new transaction boundaries.

Parameters Considered in the Classification Process

Based on Web Page Characteristics. A content page is a page with information that could be of interest to a user as he browses a web site. The size of a web page can be considered equal to the number of terms found on the page. The larger the page size, the more likely it is that the page is a content page. A web page also typically has one or more hyperlinks. In general, the larger the number of hyperlinks on a web page, the more likely it is that the page is a navigation page. Because web pages usually comprise text and hyperlinks, most web pages exhibit both navigation and content properties, and one or more metrics other than page size and the number of hyperlinks present, are required to correctly classify them. Besides, other factors may influence the classification of web pages (for example, the links to word count ratio, LW). Intuitively, the larger the value of LW, the more likely it is that the page is a navigation page, and vice versa for content pages. It should be noted though, that some web pages with high LW values could be content rich; likewise, some pages with low LW values may have too little content.

Based on Web Site Topology. In a navigation task hyperlink labels serve as navigation aids, and as cues (or scent) to the pages they represent. These cues can be exploited to determine pages in a web site that can be treated as important and thus recommendable. In this research, we measured information scent using a variation of Google's *PageRank* algorithm.

PageRank *PR*, (Craven, 2003; Rogers, 2003) is a number that Google[20] uses to determine the importance of a web page, and is used as one of several parameters to determine the ranking of pages returned by the Google search engine. The PageRank algorithm assumes that a hyperlink from one page to another is a vote from the former to the latter. The more votes a page receives, the more important that page is assumed to be. Also, the importance of a vote cast on a page is dependent on the importance of the page from which the hyperlink originates. The following equation is used to determine the page rank of *Page A*:

$$PR(A) = (1-d) + d(PR(T_1)/C(T_1) + ... + PR(T_n)/C(T_n)) \qquad (1)$$

where, $PR(T_I)$ is the rank of page *I*; *C(TI)* is the number of hyperlinks on page *I*; $PR(T_I)/C(T_I)$ is the contribution of page *I* to the rank of page *A*, if there is a link from page *I* to page *A*; and *d* is a damping factor, usually set to 0.85. PageRank can be computed through an iterative process, with the PageRanks on the right hand side of the equation initialized to any value; after a number of iterations, the page rank of each page converges.

The problem with PageRank in the domain of web navigation is that it is based solely on the network of links between web pages; it does not predict correctly how real users navigate the website, and it says nothing about the content or size of pages. Hence, a page with several links pointing to it is likely going to be judged important, even if its content is minimal (and hence, not recommendable), while a page with several outgoing links is likely to lose much of its page rank, even if its content is important. For example, the index page of a website usually has very many incoming links, and a large PageRank, even though the user is unlikely to be particularly interested in the contents of that page.

Combining PR and LW. One way to improve usefulness of PageRank in determining the importance of a web page in a navigation task is to combine it with information quantifying the contents of, and the number of hyperlinks on the page. We use the intuition that a page with many incoming links, but with little content is not as worth recommending, even if its page rank value is high, as a page with very few incoming links, but with much content, even if its page rank is low. As mentioned above on Page 51, a plausible categorization of web pages as navigation or content can be based on a comparison between the term count of the page and the number of outgoing hyperlinks on the page. Let *LW* represent the *links-to-Word count* ratio (i.e., number of outgoing hyperlinks/term count). Let ILW represent the

[20] www.google.com

Outgoing Links Count	Incoming Links Count	LW	Content Rating (ILW)	PageRank (PR)	PR × ILW compared to PR
L	L	L	H	M	M_{++}
L	L	H	L	M	M_{+}
L	H	L	H	H	H_{++}
L	H	H	L	H	H_{+}
H	L	L	H	L	L_{++}
H	L	H	L	L	L_{+}
H	H	L	H	M	M_{++}
H	H	H	L	M	M_{+}

L = Low ; H = High; M = Moderate; + = small increase; ++ = large increase

Figure 12. Illustrating the effect of ILW on PR in the PR × ILW metric.

inverse links-to-word count ratio (i.e.,1/LW).

Figure *12* suggests that PR × ILW is a better metric to use to determine the content rating (how likely it is that a page is a content page) of a page than PR alone, since it uses evidence from two sources: LW and PR, to determine content pages. For example, the third row of the figure represents pages with low LW values (i.e., high content rating), and large incoming/outgoing links ratio (i.e., high PR). The corresponding PR × ILW is equivalent to a large increase in the PR score from its current large value, since both PR and ILW are *large*, i.e., both page parameters and website topology suggest that the page is important. On the other hand, for the fourth row, which corresponds to web pages with high PR and low content rating, there is only a *small* increase in PR from its current large value since one parameter (LW) suggests that the page is not a content page while PR suggests that it is an important page. A similar argument can be made for all the other rows of the figure.

3.2.2. Association Rule Mining Objectives

Data mining—or *knowledge discovery from databases* (KDD)—aims to find useful patterns in large databases. Association rule mining is a data mining technique that finds interesting correlation relationships among a large set of data items (Agrawal, Imielinski, & Swami, 1993). Association rules that describe the correlation relationships between typical past user navigation paths and corresponding pages that they found interesting can be mined from user access logs, and these rules used to compute prediction models for current user interests. The main ideas in association rules mining are presented below.

An association rule is an implication with a *support* value, and a *confidence* value. For example, the association rule:

$$\text{buys}(X,\text{"bread"})\quad \text{buys}(X,\text{"cereals"})\qquad \text{buys}(X,\text{"milk"})$$
$$[\text{support}\quad 1\%, \text{confidence}\quad 70\%]$$

states that a person who buys bread and cereals has a 70% chance of also buying milk, and this rule is valid for 1% of the entries in the database of transactions. Support and confidence are rule interestingness measures, respectively of usefulness and certainty of the rule. Support is the percentage of items in the rule that occur together in the set of transactions. Confidence is the percentage of the rule antecedent that leads to the consequent. Minimum support and confidence levels can be set at the start of the mining process to prune out uninteresting patterns, and in the process, reduce the computational load on the system.

Let the database of items (e.g., goods in a shop, web page URLs, etc.) to be mined be referred to as the transactions database, *J*. Let the set of items in *J* be represented by $\{i_1, i_2, \ldots, i_n\}$. Let *D* be the set of database transactions $(D \subseteq J)$ that are being mined (e.g., monthly sales, URLs visited in a month, etc.). Let *X* be a set of items (e.g., items in a shopping basket, URLs representing a user's browsing behavior, etc.). *X* is referred to as an *itemset*. A *transaction, T* is said to contain *X* if $X \subseteq T$.

An association rule is an implication of the form $A \Rightarrow B$ where $A \subset J$, $B \subset J$, and $A \cap B = \phi$. The support of the rule is the probability, $P(A \cup B)$, i.e., percentage of the transactions in *D* that contain both itemsets *A* and *B*. The confidence of the rule is the conditional probability, $P(B/A)$, i.e., the percentage of transactions in *D* containing *A* that also contain *B*.

An itemset comprising *k* items is referred to as a *k-itemset*. The occurrence frequency of an itemset, also known as the *support count*, is the number of transactions in *D* that contain the itemset. An itemset is considered to be *frequent* if its support count is greater than a predefined *minimum support count*. The objective of association rule mining is to determine correlations between items in frequent itemsets. The discovered rules are considered strong or interesting if they meet both the minimum support count and a minimum confidence level.

The *apriori property* is useful in association rules mining; it states that all nonempty subsets of a frequent itemset must also be frequent, and is based on the observation that if an item *A* is added to a non-frequent itemset *I*, the new itemset $(A \cup I)$ cannot be more frequent than *I*. Another way of saying this is that if $(A \cup I)$ is frequent, then both *A* and *I* must necessarily be frequent for, if *A* or *I* were infrequent, then $(A \cup I)$ would have been infrequent. The apriori property is an example of an anti-monotone property with the characteristic that if a set cannot pass a test, then all of its supersets will fail the same test (Han & Kamber, 2001).

The *apriori algorithm* (Aggarwal & Srikant, 1994; Han & Kamber, 2001; Mannila, Toivonen, & Verkamo, 1994) has been influential in association rule mining (Han & Kamber, 2001). The algorithm makes use of the apriori property to reduce the search space required to determine candidate itemsets for inclusion in the association rule set. The classical apriori algorithm involves a *join* in each of *n* transactions database scans, to determine candidate itemsets for inclusion in the association rules set. There are two main drawbacks to the classical apriori algorithm: (1) the number of candidate itemsets generated by the join step may be too large to fit into main memory; and (2) high latency which results from the need to scan the database during each iteration. Any algorithm that tackles any of these problems is likely to lead to significant improvement on the efficiency of the mining process.

The two problems mentioned above were investigated in this study, resulting in the development of a new, *joinless apriori* algorithm that, in general, significantly reduces the search space required to determine frequent itemsets. The new algorithm eliminates the join step in the classical algorithm, and applies the apriori property to each record in the transactions database.

3.2.3. Construction of Prediction Models for Current User's Interests

Two approaches for the prediction of what web pages users with a given navigation behavior would be interested in, both based on collaborative filtering, were investigated. Collaborative Filtering Model I used the center-based scheme and Model II, the aggregate-based scheme in determining the neighborhood vectors used to compute the predictions.

The starting point in our approach to building the prediction model is previous research by Mobasher and colleagues (Mobasher et al., 2000). They described an approach to computing hyperlink recommendations from association rules mined from web server logs, which involved *sliding* the current user's page accesses through a fix sized window, using the window to represent user's current navigation behavior, and computing a recommendation score for a user's navigation behavior if the corresponding window corresponded to the antecedent of one or more association rules. The process involved is illustrated in Figure 13. The figure shows (a) sample association rules depicting user navigation behaviors for two pages of interest X and Y; (b) a sample user session; and (c) the different states of the user behavior vector as the user browses the website. Assuming the window size is 4, then only State 8 of the window triggers the computation of a new set of recommendation scores for pages X and Y because that is the only state that corresponds to one or more association rule antecedents (shown shaded in Figure 13 (a)). Recommendation scores for pages X and Y would then be computed as follows: $r_x = c_x \times ldf_x$ and $r_y = c_y \times ldf_y$ respectively, where c is the confidence of the association rule and ldf (link distance factor[21]) is a number that is inversely proportional to the link distance (i.e., number of hops) from the current

[21] The overall effect of link distance factor is to boost the recommendation score of high confidence pages that are *far* away from the current navigation window, thus increasing their chances of making it to the recommendation list.

(a)

Sample Association Rules

A ⇒ X	A ⇒ Y
B ⇒ X	B ⇒ Y
C ⇒ X	C ⇒ Y
AB ⇒ X	AB ⇒ Y
AC ⇒ X	AC ⇒ Y
BC ⇒ X	BC ⇒ Y
ABC ⇒ X	ABC ⇒ Y
ABD ⇒ X	ABD ⇒ Y
ABE ⇒ X	ABE ⇒ Y
ABF ⇒ X	ABF ⇒ Y
BCD ⇒ X	BCD ⇒ Y
BCE ⇒ X	BCE ⇒ Y
BCF ⇒ X	BCF ⇒ Y
CDF ⇒ X	CDF ⇒ Y
ABCD ⇒ X	ABCD ⇒ Y
ABCE ⇒ X	ABCF ⇒ Y
BCDF ⇒ X	BCEF ⇒ Y
⋮	⋮

(b)

Sample user navigation behavior

A	B	C	A	B	C	D	P	…

(c)

Navigation Window States
(Window size = 4)

1	----	
2	---A	
3	--AB	
4	-ABC	◘
5	ABCA	◘
6	BCAB	◘
7	CABC	◘
8	ABCD	☺
9	BCDP	

◘ Very similar to one or more frequent itemset(s)
☺ Exact match with a frequent itemset

Figure 13. Illustrating Mobasher et al.'s (2000) approach to computation of hyperlink recommendations from association rules: (a) sample association rules; (b) sample user session; (c) changes in the state of the user behavior vector. The highlighted association rules are the only ones that matche any of the user behavior vectors.

navigation window, to the recommendable page. Since this research is restricted to the off-line computations involved in recommender system design, we do not consider the computation of recommendation scores any further.

We make the following observations from the approach used by Mobasher et al. (2000). First, only association rules whose antecedents match the user's current behavior vector contribute to the computation of recommendation scores. Second, there are a number of similar navigation behaviors exhibited by users that do not contribute to the recommendation score (e.g., states 4, 5, 6, and 7 in Figure 13(c), each of which has only one page different from one or more association rules). It is desirable to have association rules of varying lengths and/or whose antecedents are very similar to the current navigation window contribute to the computation of recommendation scores, especially if these rules have high confidence values.

This work tackled the shortcomings mentioned above by converting the recommendation process to a collaborative filtering problem in which user navigational behaviors represented on the left hand side of the mined association rules are analogous to users in a classical collaborative filtering problem, and the confidence values of their likeness for the pages on the right hand side of the rules are analogous to user ratings for items. When a user interacts with the system, the similarity between his navigation window to the association rule antecedents could then be computed, and the most similar rules (i.e., nearest neighbors) selected.

Collaborative filtering techniques could then be used to predict the confidence scores for web pages that could be recommended to the user, and recommendation scores computed for these pages. Our expectation was that the system would benefit from the numerous advantages of collaborative filtering (see Section 2.2.2.1, Page 9), and at the same time, avoid the following problems typical in collaborative filtering systems: the sparsity, new user, and new item problems (see Section 2.2.2.1, Page 9). These problems were expected to be avoided because the database of user transactions vs. pages of interest would already be populated with confidence values from association rule mining, and it is the exceptional user whose navigation pattern will not be similar to one of those already mined.

3.2.4. Clustering Objectives

Clustering is commonly used in the design of some offline components of recommender and other personalization systems. Traditional approaches to clustering include partitioning methods like k-means and k-medoids, and hierarchical approaches. One characteristic that limits the utility of these approaches is the fact that they are only efficient when dealing with small amounts of data, typically with very few dimensions. Recently though, researchers have devised new approaches that deal with larger data sets. Examples include partitioning approaches like CLARA (Kaufman & Rousseeuw, 1990) and CLARANS (Ng & Han, 1994, 2002); hierarchical methods like CURE (Guha, Rastogi, & Shim, 1998) and CHAMELEON (Karypis, Han, & Kumar, 1999); integration of iterative and hierarchical approaches like BIRCH (Zhang, Ramakrishnan, & Livny, 1996); density-based methods like DBSCAN (Sander, Ester, Kriegel, & Xu, 1998); and grid-based methods like STING (Wei, Yang, & Muntz, 1997). Most of these works have however, been limited to low-, (typically 2-) dimensional data.

Ng and Han (1994; 2002) proposed CLARANS as a memory-based approach to effectively cluster large amounts of spatial data. These data are commonly represented with a series of data points in 2- and sometimes 3-dimensional space, with the similarity measures between objects in the space usually determined from Euclidean distances between them. By simulating high dimensional document spaces, we sought to determine if CLARANS can be extended to generate clusters from much higher dimensional data using a non-Euclidean measure (the angle between document vectors) for the dissimilarity measure. The need of a non-Euclidian measure for dissimilarity resulted from the inadequacy of Euclidean measures in high dimensional space, where vectors are not very distinguishable from each other. If CLARANS turned out to perform well in high dimensional space, them it would be an attractive clustering algorithm in IR and related domains like hyperlink recommender systems design, in which data (documents) are usually represented in high dimensional space.

4. IMPLEMENTATION

This section presents the work that was done in relation to the problems identified in Section 3. The material is presented in the following order: determination of website topology and content (Section 4.1); generation of transaction logs (Section 4.2); discovery of association rules from transaction logs (Section 4.3); design and application of CLARANS clustering algorithm (Section 4.4); and design of prediction models for current user interests (Section 4.5).

4.1. Determination of Website Topology and Content

Knowing the topology of the website is important for at least two reasons: (1) it helps in one of the heuristics used in determining pasts user navigation paths by starting a new path if a page access is found that is not reachable by hyperlink from any of the currently visited pages; and (2) the site topology may be useful at recommendation time by favoring those recommendable pages that are *far* from the user's current navigation path. Determination of the topology of the website and contents of the HTML pages involved discovering the web pages and their contents, and the site's link structure. The process involved two steps: (1) design and use of a web crawler to discover, parse and code the HTML pages of the website and (2) construction of a graph structure from the discovered HTML pages, to represent the site's topology.

4.1.1. Discovery and Parsing of Web Pages Found in the Site

A crawler was designed to search for, and parse the HTML pages in the website. The crawler was a breadth-first search program that iteratively performed the following tasks on the HTML pages it found, starting at index page of the website.

1. Get the URL of the next static HTML page in the queue of pages awaiting parsing
2. Make an HTTP connection, and read the HTML page
3. Extract the individual words on the current HTML page (representing page contents)
4. Parse the page to determine outgoing hyperlinks to other HTML pages
5. Add all outgoing hyperlinks that have not yet been parsed and that referenced pages internal to the site, to the end of a queue of pages awaiting parsing
6. Code the current page and store its contents

It should be noted that there was no guarantee that the crawler would find all the HTML pages of the website. That would be the case for example, if there were pages that were not referenced from any other page in the website. Likewise, an HTML page would not be found if it were buried several layers deep such that the crawler terminated before it had the opportunity to reach the page. In spite of these shortcomings, most of the HTML pages in the website were expected to be found.

We next detail the steps that were followed to accomplish Steps 3 and 4 above.

Extraction of Page Contents from Current HTML Page. HTML pages comprise not only the content the user sees, but also hidden code whose primary purpose is to help with presentation of the contents. It was necessary to filter out these presentation-related codes to get a more accurate representation of page contents from the point of view of the user. The following steps were involved:

- Define a stop list (see Appendix B) to filter out HTML tags
- Parse the terms between the HTML body start and end tags, and use the following heuristics to determine if a term should be kept or ignored
    - Reject everything between "<!--" and "-->", since these denote HTML comments or introduce scripts, which have nothing to do with page content
    - Reject terms that begin with '&' and end with ';'; they denote special characters
    - Reject terms between "<" and ">", if the first term after "<" is in the HTML stop list
    - Ignore pages with long character sequences with no white space or tags; these usually indicate the use of some foreign alphabet, not considered in this study.

The steps outlined below were used to reduce the dimensionality of the HTML document space

- Use of the Porter stemming algorithm (Porter, 1980) to represent several words with the same stem as the same
- Use of SMART stoplist (Rocchio, 1971) to remove very commonly used words in the English language, which as a consequence do not help in distinguishing documents in which these words are used
- Elimination of low and high frequency terms by applying the following heuristics:
    - Consider all terms whose occurrence frequency is less than 1% of the total number of HTML pages available in the web site as low frequency terms
    - Consider all terms whose occurrence frequency is more than 25% of the total number of HTML pages available in the web site as high frequency terms

Finally, the vector space (or TFIDF) model was used to represent each web page. Equation 2 was used to assign term weights to the index terms of the web page collection.

$$w_{ik} = f_{ik}(\log_2(\tfrac{N}{D_k}) + 1) \tag{2}$$

where,

$$\begin{aligned} w_{ik} &= \text{weight of term k in document i} \\ f_{ik} &= \text{frequency of term k in document i} \\ N &= \text{number of documents in the collection} \\ D_k &= \text{number of documents containing term k} \end{aligned}$$

We notice that the assigned weight for each index term is proportional to its frequency in that document, but inversely proportional to its frequency across all available documents as required by the model. It is also necessary to avoid giving preference to longer documents in which any given term is likely to occur more frequently than in shorter documents, and so document lengths need to be normalized. Normalization comes as a side effect when document similarities are computed using the cosine similarity measure, since the document lengths appear at the denominator of the cosine similarity metric (see Appendix A). Logarithm was used in Equation 2 because it is insensitive to large variations in *N* and *D*.

Determination of Hyperlinks from Current HTML Page. The following steps were used to determine valid hyperlinks from a current HTML page:

- Search for anchor (<A> and </A>) and area (<AREA> and </AREA>) HTML tags, and the corresponding URLs preceded by an HREF tag
- Ignore the URL if the corresponding page is hosted by a server other than the server of the web site of interest (absolute addresses of web pages of interest were assumed to be preceded by "http://www.sis.pitt.edu" or "http://www2.sis.pitt.edu").
- Ignore the URL if it references a bookmark within the current page, to avoid duplicate treatment of the same page
- Ignore the URL if its name did not have an *htm* or *html* extension
- Normalize file names to avoid treating the same file with different name representations as different. The following two normalization steps were implemented:
  - Convert abbreviated path names to full path names, for example, path names of the form *abc/def/../ghi* were changed to the equivalent *abc/ghi* form.
  - Convert the commonly used "%7E" and "%7e" escape sequences in file names to corresponding character, "~".

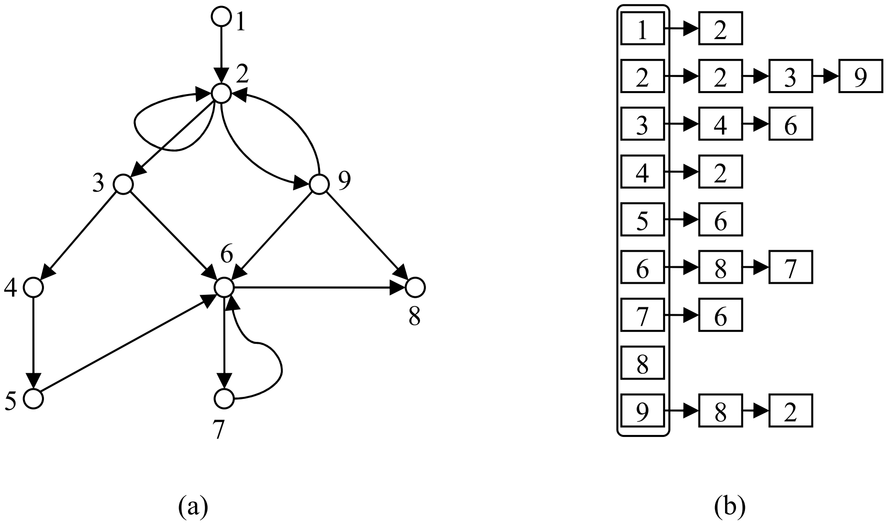


Figure 14. Illustrating adjacency list representation of website topology. (a) sample website; (b) corresponding adjacency list representation.

4.1.2. Web Site Topology

The HTML pages found by the crawler were used to construct the website's topology, which was coded as a graph that had vertices as web pages (nodes), and edges as directed hyperlinks. For each node, the directed hyperlinks were simply the outgoing links that were found on the pages as described in in Section 4.1.1 above.

The graph was constructed using an adjacency-lists representation. The representation comprised an array of linked lists, with each array index representing a vertex, and the entries in each linked list, the nodes that the current vertex linked to. Figure 14 illustrates the representation used.

4.2. Generation of Transactions Log

This section describes the approach that was followed to derive the transactions log (past user behavior data) from the user access log file. The user access log data used were structured in the *common log format* (W3C, 2003), which comprises the following fields for every record:

remotehost: Remote hostname (or IP address number if DNS hostname is not available or was not provided);

rfc931: The remote login name of the user (if not available a minus sign is typically placed in the field)**;**

authuser: The username with which the user has authenticated himself. This is available when using password protected WWW pages (if not available a minus sign is typically placed in the field);

[date]: Date and time of the request.;

"request": The request line exactly as it came from the client (i.e., the file name, and the method used to retrieve it (typically GET));

status: The HTTP response code returned to the client; it indicates whether or not the file was successfully retrieved, and if not, what error message was returned;

bytes: The number of bytes transferred.

When a user request reaches a web server, the web server records a *page view* (Mobasher et al., 2000), comprising the requested page and if present, other files (e.g. graphic and sound files embedded in the page). This happens because the HTTP protocol requires a separate request for each of these files embedded in the requested page. Because a web server receives and logs several user requests simultaneously, one user's session data are typically interspersed with those of other users. Generation of the transaction logs involves extracting individual user sessions from the access logs, and transactions within individual user sessions. The only fields of interest in this work were *remotehost*, *date*, and *request.* The rest of this section shows how the access log data were cleaned to obtain the required data, discusses the application of heuristics to determine user sessions from the data, and transactions within the sessions, and presents an example to illustrate the processes involved.

4.2.1. Data Cleaning

The main objective of data cleaning was to obtain data that comprised only the pages that were explicitly requested by the user, by deleting the additional entries in the server logs that corresponded to files embedded within the requested pages. These additional files needed to be filtered out lest they introduced noise to the transactions log, and subsequently to the mined association rules.

The first stage of data cleaning involved the removal of all records of non-HTML pages. The second stage of data cleaning involved deleting all records from the access logs that represented single page accesses (i.e., the IP address only appeared once) since such data were unlikely to represent valid user browsing, and since the objective was to eventually generate association rules that showed the relationship between at least two pages: one or more navigation page(s) and a content page.

4.2.2. Session Identification

A session comprises the complete history of web pages that a user visits each time he visits a website. Session identification was the first step in the discovery of user navigation behaviors. The process of identifying user sessions was complicated by two problems: caching at user browsers, proxy servers and firewalls (see Section 3.1), which resulted in some user requests not reaching the web server, and the absence of user IDs from the web logs.

The following steps were used to identify user sessions.

- Sorting of the cleaned access logs in order of IP address and order in which the requests were recorded
- Use of the *remotehost* field (i.e., IP address) of the request as a surrogate for user ID. This was only a first step since IP addresses do not uniquely identify the source of the requested page, as described in Section 3.1.
- Requests from the same IP address typically spanned several hours, days, or even weeks partly because the IP address represented different users at different times, and/or the same user at different sessions. The following heuristic was used to break user sessions into manageable lengths: a session was assumed to last no more than 30 minutes. Hence, for the same IP address, a new user session was started at the next occurrence of the IP address if the time of the later access was more than 30 minutes after the first occurrence of the same IP address in the current session.
- To help distinguish different users who shared the same IP address within the same 30-minute (or less) block of requests, the following heuristic was used: if a web page was accessed that was not directly reachable[22] from any of the pages previously visited by any of the users sharing the same IP address, then that page access was assumed to come from a different user, and a new session started for that user.

  This last heuristic would fail if a user accessed a URL that was not in the list of already accessed URLs, by typing it out rather than by following a hyperlink. But this is not a typical browsing behavior, and only a small percentage of such cases were expected to be found in the access log file.

  If a page that was directly reachable from more than one session with the same IP address was accessed, then a least recently used (LRU) policy was used to attribute that page access to the user who had the least recent navigation action attributed to him. This situation

[22] A web page is reachable from another web page if there is a hyperlink from the former to the latter. A hyperlink is represented by a directed edge on the graph representing the website.

was expected to occur only when several users sharing the same IP address were browsing a similar set of pages at the same time. This was not expected to occur frequently, and so the errors resulting from this heuristic were not expected to be severe.

4.2.3. Transaction Identification

A user session comprises all the pages that a user accesses during a single visit to a website; a transaction on the other hand, is a complete logical unit within a session comprising a set of navigation pages followed by a corresponding content page (assumed to be a page of interest to the user). In this study, each transaction was used to represent the user's changing navigation behavior at that stage of his browsing activity.

Having discovered individual user sessions from the server logs, the next task was to discover individual transactions within user sessions. Two heuristics were used to determine transactions. The first heuristic made use of the notion of *maximal forward reference, MFR*, (Chen et al., 1996). During web browsing, the MFR is the last page that is accessed before the user requests a page previously viewed during that session. All MFR pages were considered to be pages of interest, while the pages before the MFR were considered to be navigation pages. After a MFR page was discovered, the next navigation path was assumed to start with the page following the MFR page.

Following the discovery of transactions using the MFR heuristic, the PR × ILW heuristic was applied to discover further transactions within transactions that had lengths greater than 5 by classifying the pages involved as content or navigation, and breaking the transaction at the first content page found. The following procedure was used to classify the pages:

- Compute PR, LW, ILW, and PR × ILW for every page of the Web site
- Have a panel of experts rate a sample of HTML pages as content or navigation
- Aggregate the experts' ratings to determine boundary values for LW and PR × ILW
- Extrapolate the results to the other web pages of the site

The study that involved rating of web pages by experts is presented in User Study 1, Section 4.2.5.

4.2.4. From Session to Transactions Data – An Example

We present an example involving a small web access log file to illustrate the process of converting web access log data to a transactions database. Assume that the access log comprises the entries shown in Figure 15, where the date has been replaced by a timestamp that shows the order in which the entries were made in the log file. Also assume Figure 16 to represent the website. We start by identifying the various user sessions present. Figure 17 shows that three sessions can be extracted for the access log in Figure 15 – sessions S001

(F, Y, A, B, C, D, C, B, D, E, F, G, H, I, J, K, L, G, L, Q) and S002 (D, E, F, C, Q, M, L, Q, M, G, L, F, C, Q, B) with the same IP address (IP1), and S003 (C, D, E, Y, A, C, Q) with a different IP address (IP2).

The following checks are made for every record of the access log:

- If a new IP address is encountered, a new session is started
- If an existing IP address is encountered but the page requested is not reachable from any of the pages previously encountered for that session, then a new session is started
- If an existing IP address is encountered and the page requested is reachable from a user session, then the page is added to that session; if more than one of the sessions share that IP address, then the page is added to the session that was least recently updated.

Date	Remote host	Requested page	Date	Remote host	Requested page	Date	Remote host	Requested page
1	IP1	F	15	IP2	Y	29	IP1	L
2	IP1	D	16	IP1	Q	30	IP1	I
3	IP1	Y	17	IP1	B	31	IP1	J
4	IP2	C	18	IP1	M	32	IP1	K
5	IP1	A	19	IP1	D	33	IP1	F
6	IP1	B	20	IP2	A	34	IP1	L
7	IP2	D	21	IP1	L	35	IP1	C
8	IP1	E	22	IP1	E	36	IP1	G
9	IP1	C	23	IP1	Q	37	IP1	Q
10	IP1	F	24	IP1	F	38	IP1	L
11	IP2	E	25	IP1	M	39	IP1	B
12	IP1	D	26	IP1	G	40	IP1	Q
13	IP1	C	27	IP1	G	41	IP2	C
14	IP1	C	28	IP1	H	42	IP2	Q

Figure 15. Sample web server access log file.

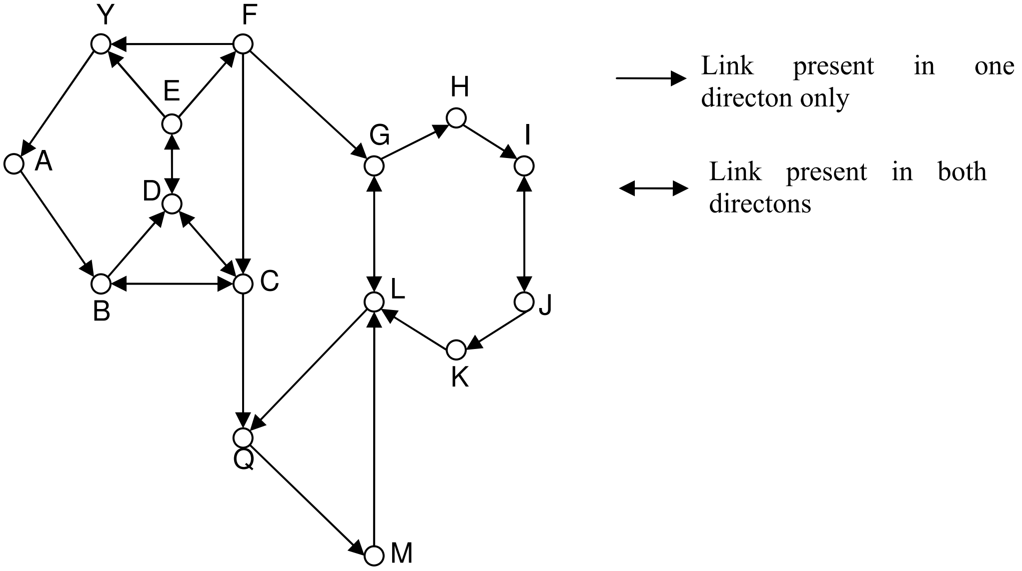

Y
F
E
H
A
G
I
D
C
B
L
J
K
Q
M
Link present in one directon only
Link present in both directons

Figure 16. Sample website used to illustrate generation of transaction logs.

Session & IP addr.	Page assignments to sessions																																									
S001 IP1	F		Y		A	B			C			D		C			B		D			E		F		G		H		I	J	K		L		G		L		Q		
S002 IP1		D						E		F			C			Q		M			L		Q		M		G		L				F		C		Q		B			
S003 IP2				C			D				E				Y					A																					C	Q
	Iteration & reason for assignment																																									
	1 I	2 N	3 R	4 I	5 R	6 R	7 R	8 R	9 L	10 L	11 R	12 L	13 L	14 L	15 R	16 L	17 L	18 R	19 L	20 R	21 R	22 L	23 L	24 L	25 R	26 L	27 L	28 L	29 L	30 R	31 R	32 R	33 L	34 L	35 L	36 L	37 L	38 L	39 L	40 L	41 R	42 R
	I: New IP addr. – create new session R: Page reachable from this session only L: Page reachable from several sessions – choose least recently updated session N: IP addr. seen before but page nor reachable from any session with that IP address – create new session																																									

Figure 17. Sessions obtained from access log in Figure 15.

With the sessions identified, the next step is to break them into transactions. Let us consider session S001: {F Y A B C D C B D E F G H I J K L G L Q}.

- The first step in the discovery of transactions within the session is to determine all maximal forward references. These are shown shaded below.

F	Y	A	B	C	D	C	B	D	E	F	G	H	I	J	K	L	G	L	Q

- Next, we break up the session into transactions at the MFR page, treating the MFR page as content (or recommendable page). Figure 18(a) shows the resulting transactions.
- Finally, we use the PR × ILW metric to determine if there are other content pages in the transactions that the MFR heuristic could not detect. Let us assume that pages *A*, *G*, and *K* are the only pages that have PR × ILW values above a threshold, meaning they are considered to be content pages. We scan the transactions and break them up further at these content pages. Figure 18(b) shows the results. In this example, the user session breaks down into ten separate transactions, with the last URL in each transaction representing a content page, and the preceding URLs, navigation pages. All other user sessions are treated in a similar manner.

S001 T001 F Y A B C D
S001 T002 C
S001 T003 B
S001 T004 D E
S001 T005 F G H I J K L
S001 T006 G
S001 T007 L Q

(a)

S001 T001 F Y A
S001 T002 B C D
S001 T008 C
S001 T003 B
S001 T004 D E
S001 T005 F G
S001 T009 H I J K
S001 T0010 L
S001 T006 G
S001 T007 L Q

(b)

Figure 18. Transactions resulting from access logs in Figure 15: (a) using MFR heuristic; (b) using PR × ILW heuristic.

4.2.5. User Study 1: Web Page Classification Study

The objective of this study was to determine threshold values for the LW and PR × ILW parameters, which determined whether web pages were classified as content or navigation pages. This classification would then determine transaction boundaries for very long transactions that the MFR heuristic alone could not break down into smaller sizes.

Subjects. The subjects comprised three users who were very familiar with the website of SIS: two fifth year and one third year Information Science graduate students.

Procedure. The HTML pages of the website and their contents were collected as described in Section 4.1.1, and the site topology determined as described in Section 4.1.2. Next, the PR, LW, and PR × ILW metrics were computed for each of the pages as follows:

PR for each page was computed using Equation 1 (Page 52). The value of the damping factor d, was set to 0.15, PR for each page initialized to $1/N$, where N was the number of web pages in the site, and the computation was repeated for 30 iterations.

LW for each page was simply the ratio of the number of hyperlinks on the page to the term count (or page size) for that page.

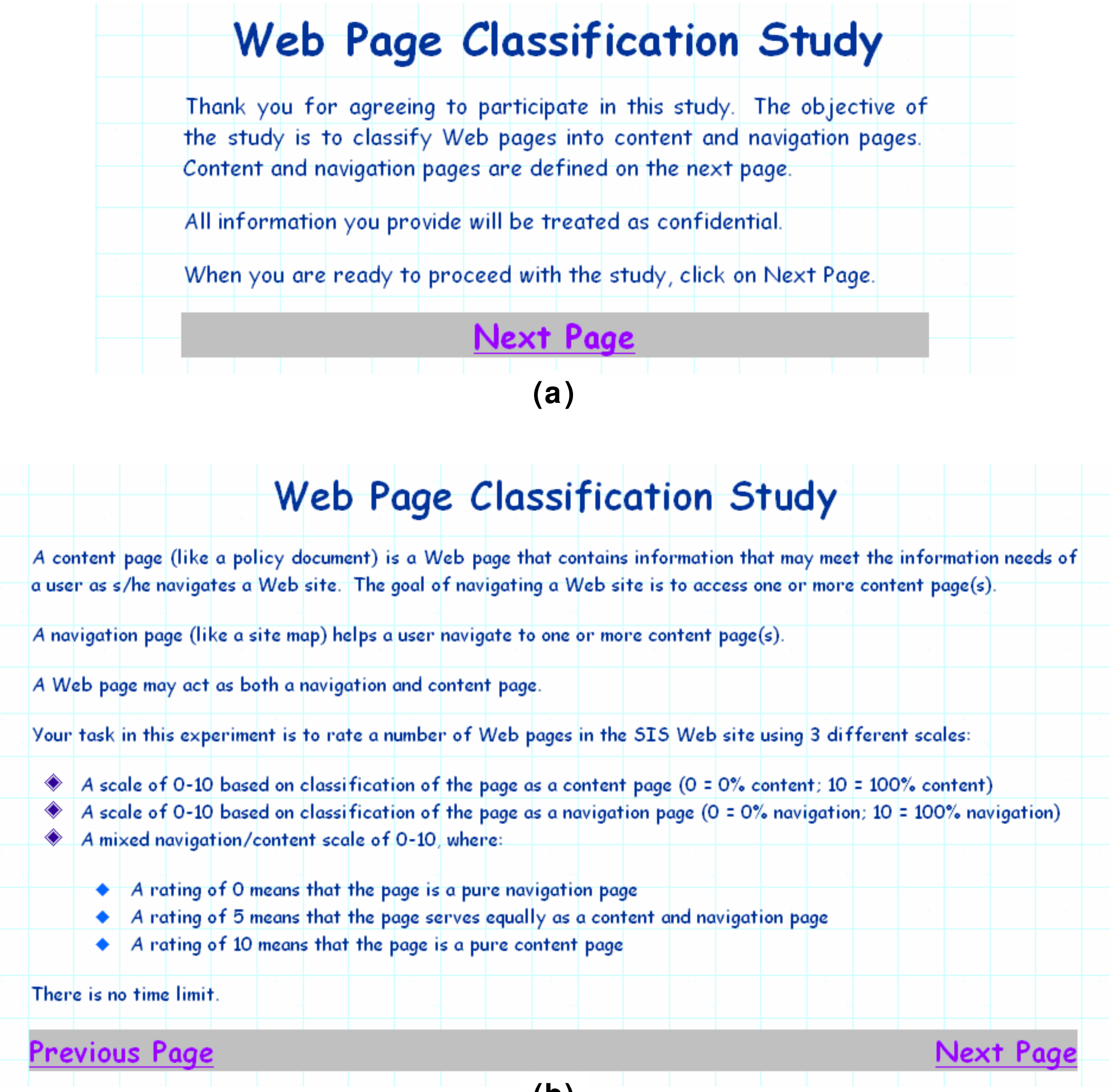
Finally PR × ILW was computed as PR/LW, with the following provision made for pages with LW value of zero: for such pages, PR × ILW was computed using the formula PR × term count for that page. That way, the following desirable goals were achieved: (1) the divide by zero error was eliminated; and (2) longer pages (higher content) were favored over shorter ones.

The URLs of 4 web pages were then randomly assigned without replacement to each of the following 9 groups: PR_{high} (PR of the page is among the top third of page ranks), PR_{med} (mid third of PR values) and PR_{low} (bottom third of PR values); ILW_{high}, ILW_{med}, ILW_{low}, for the ILW metric; and $\text{PR} \times \text{ILW}_{\text{high}}$, $\text{PR} \times \text{ILW}_{\text{med}}$, $\text{PR} \times \text{ILW}_{\text{low}}$, for the PR × ILW metric. Finally, all 36 randomly assigned URLs were presented as hyperlinks on a web page to the subjects, who were instructed to view the corresponding web pages and rate them on a scale of 0 to 10 using three different scales: a content scale, a navigation scale, and a dual content/navigation scale. Subjects responses were collected using an HTML form.

Figure 19 is a summary of the research design. Figure 20 shows the instructions that each subject received, and Figure 21 a screenshot of one of the rating screens presented them.

Users	Page characteristics								
	PR_{high}	PR_{med}	PR_{low}	$\text{PR} \times \text{ILW}_{\text{high}}$	$\text{PR} \times \text{ILW}_{\text{med}}$	$\text{PR} \times \text{ILW}_{\text{low}}$	LW_{high}	LW_{med}	LW_{low}
1	User ratings of web pages using content, navigation, and dual content/navigation scales								
2									
3									

Figure 19. Research design for the web page classification study.



Web Page Classification Study

Thank you for agreeing to participate in this study. The objective of the study is to classify Web pages into content and navigation pages. Content and navigation pages are defined on the next page.

All information you provide will be treated as confidential.

When you are ready to proceed with the study, click on Next Page.

(a)

Web Page Classification Study

A content page (like a policy document) is a Web page that contains information that may meet the information needs of a user as s/he navigates a Web site. The goal of navigating a Web site is to access one or more content page(s).

A navigation page (like a site map) helps a user navigate to one or more content page(s).

A Web page may act as both a navigation and content page.

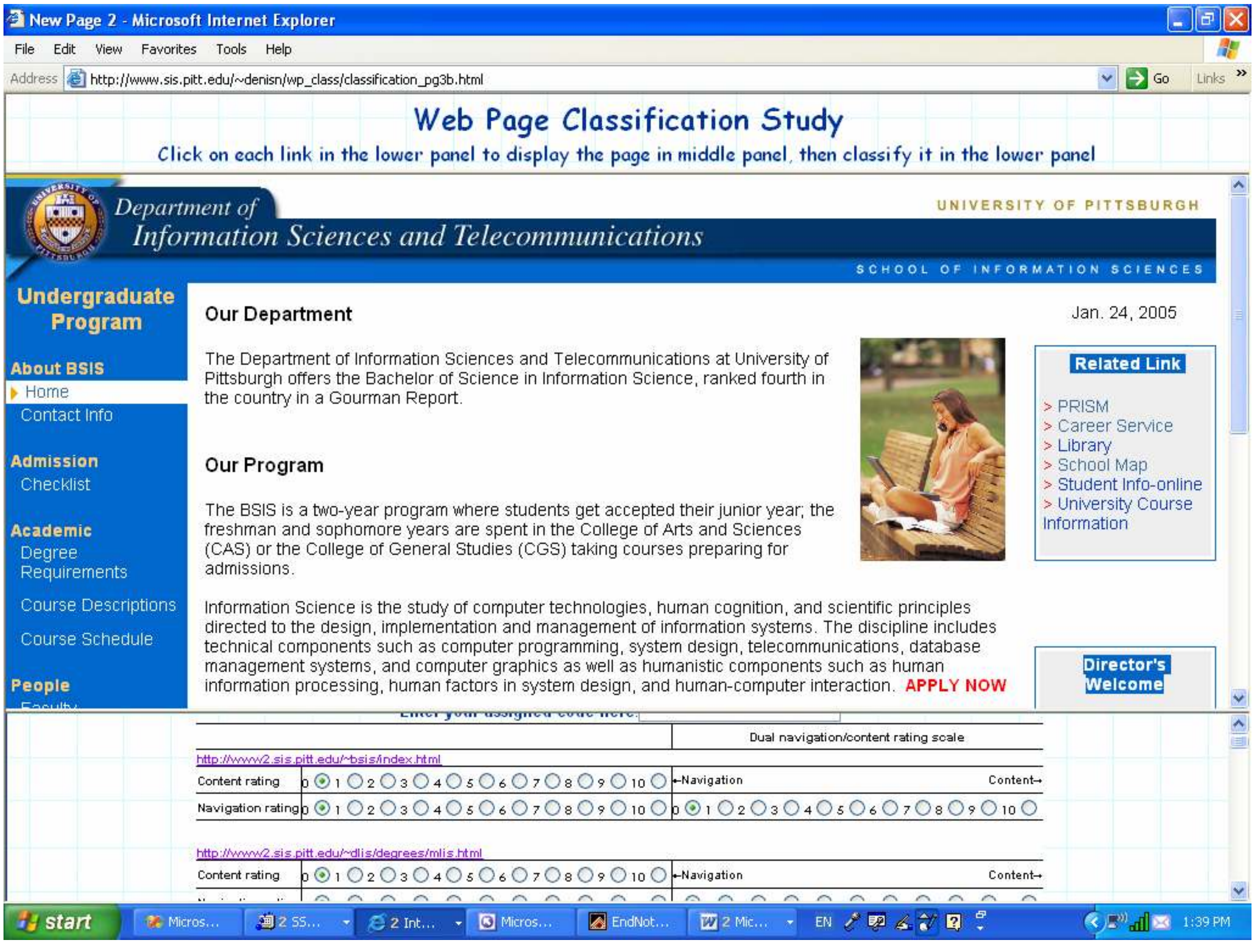
Your task in this experiment is to rate a number of Web pages in the SIS Web site using 3 different scales:

- A scale of 0-10 based on classification of the page as a content page (0 = 0% content; 10 = 100% content)
- A scale of 0-10 based on classification of the page as a navigation page (0 = 0% navigation; 10 = 100% navigation)
- A mixed navigation/content scale of 0-10, where:
  - A rating of 0 means that the page is a pure navigation page
  - A rating of 5 means that the page serves equally as a content and navigation page
  - A rating of 10 means that the page is a pure content page

There is no time limit.

(b)

Figure 20. Subject instructions for web page classification study: (a) welcome page; (b) description of task.


New Page 2 - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://www.sis.pitt.edu/~denisn/wp_class/classification_pg3b.html Go Links

Web Page Classification Study

Click on each link in the lower panel to display the page in middle panel, then classify it in the lower panel

Department of Information Sciences and Telecommunications

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

Undergraduate Program

About BSIS
Home
Contact Info

Admission
Checklist

Academic
Degree Requirements
Course Descriptions
Course Schedule

People

Our Department

Jan. 24, 2005

The Department of Information Sciences and Telecommunications at University of Pittsburgh offers the Bachelor of Science in Information Science, ranked fourth in the country in a Gourman Report.

Our Program

The BSIS is a two-year program where students get accepted their junior year; the freshman and sophomore years are spent in the College of Arts and Sciences (CAS) or the College of General Studies (CGS) taking courses preparing for admissions.

Information Science is the study of computer technologies, human cognition, and scientific principles directed to the design, implementation and management of information systems. The discipline includes technical components such as computer programming, system design, telecommunications, database management systems, and computer graphics as well as humanistic components such as human information processing, human factors in system design, and human-computer interaction. APPLY NOW

Related Link
> PRISM
> Career Service
> Library
> School Map
> Student Info-online
> University Course Information

Director's Welcome

Dual navigation/content rating scale

http://www2.sis.pitt.edu/~bsis/index.html

Content rating 0 ● 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10 ○ ←Navigation Content→

Navigation rating 0 ● 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10 ○ 0 ● 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10 ○

http://www2.sis.pitt.edu/~dlis/degrees/mlis.html

Content rating 0 ● 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10 ○ ←Navigation Content→

start Micros... 2 SS... 2 Int... Micros... EndNot... 2 Mic... EN 1:39 PM

Figure 21. Screenshot of one of the web pages that subjects were required to classify as content or navigation. The top panel contains directions to subjects, the middle panel, the page currently being classified, and the bottom panel, page URLs and their ratings.

4.3. Discovery of Association Rules from Transaction Logs

As discussed in Section 3.2.2, a new joinless apriori algorithm was investigated in this study to help tackle the space and time complexity problems of the classical apriori algorithm. This section presents the joinless apriori algorithm (Nkweteyim & Hirtle, 2005), and its application to the mining of association rules from transaction logs. To help with the understanding of the procedure, we start with a description of the implementation of the classical apriori algorithm and the problems inherent in it (Section 4.3.1). We then present the new algorithm and the procedure used to tackle the problems of the classical algorithm, and demonstrate that both algorithms lead to the same mining results (Section 4.3.2). Finally, we describe the peculiar nature of the transaction logs that were mined, and describe the implementation of the new algorithm to those data (Section 4.3.3).

4.3.1. The Classical Apriori Algorithm

The apriori algorithm uses *prior knowledge* (the apriori property) of frequent k-itemsets to prune the search space for the mining of (k+1)-itemsets. It begins by finding frequent 1-itemsets, L_1, during the first iteration of the algorithm, and L_1 used to find frequent 2-itemsets, L_2, in the second iteration, L_2 used to determine L_3, in the third, etc. Every iteration of the algorithm involves a join and prune steps. The join step is used to determine frequent k-itemsets (L_k) from frequent (k-1)-itemsets (L_{k-1}) by joining L_{k-1} to itself to generate candidate k-itemsets (C_k), which, being a superset of L_{k-1}, may or may not be frequent. The prune step scans the transactions database to determine which of the itemsets (if any) in C_k are frequent. These frequent itemsets form the members of L_k.

The join step is performed as follows. First, the items within a transaction or itemset are assumed to be sorted in lexicographic order. The itemsets l_1, l_2, ... l_{k-1} of L_{k-1} are then joined to each other, where two itemsets l_I, l_J of L_{k-1} are joinable if their first k-2 items are common (i.e. $(l_I[1] = l_J[1]) \wedge (l_I[2] = l_J[2]) \wedge \ldots \wedge (l_I[k\text{-}3] = l_J[k\text{-}3]) \wedge (l_I[k\text{-}2] = l_J[k\text{-}2]) \wedge (l_I[k\text{-}1] < l_J[k\text{-}1])$, where $l_i[j]$ is the jth item in itemset l_i). The condition $l_I[k\text{-}1] < l_J[k\text{-}1]$ ensures that no duplicates occur. The result of joining itemsets l_I and l_J is $(l_I[1]l_I[2] \ldots l_I[k\text{-}3]l_I[k\text{-}2]l_I[k\text{-}1]l_J[k\text{-}1])$.

To illustrate the working of the classical apriori algorithm, let the database D, of transactions be as represented in Figure 22, which shows transaction IDs and corresponding transactions. Let the minimum support count be 3. The steps required to generate all frequent itemsets are illustrated in Figure 23. As can be seen from the figure, the algorithm iterates through the following steps:

- Determine candidate k-itemsets, C_k
- Scan the database to determine support count for each of the itemsets in C_k
- Compare the support count for each itemset in C_k with minimum support count, to determine frequent itemsets L_k
- Join L_k to itself and apply the apriori property to determine C_{k+1}; C_{k+1} may or may not contain non-frequent itemsets, but the apriori property ensures that it only contains itemsets with frequent subsequences.

TID	Transaction Items
T001	A,B,C,E
T002	B,C
T003	A,B,D
T004	A,C
T005	B,C
T006	A,B,C
T007	A,B,C,E
T008	A,B,E

Figure 22. Sample transactions database, D, for illustration of the classical and joinless apriori algorithms.

<table>
<tr><td colspan="8">Step 1a: Scan database <i>D</i> for count of each candidate 1-itemset C_1
Step 1b: Compare C_1 itemsets with minimum support & generate L_1</td></tr>
<tr><td colspan="2" rowspan="6"></td><td rowspan="6">Scan <i>D</i> to get candidate 1-itemset counts</td><td>C_1 Itemset</td><td>Support count</td><td rowspan="6">Compare C_1 support count with minimum support</td><td>L_1 Itemset</td><td>Support count</td></tr>
<tr><td>{A}</td><td>6</td><td>{A}</td><td>6</td></tr>
<tr><td>{B}</td><td>7</td><td>{B}</td><td>7</td></tr>
<tr><td>{C}</td><td>6</td><td>{C}</td><td>6</td></tr>
<tr><td>{D}</td><td>1</td><td>{E}</td><td>3</td></tr>
<tr><td>{E}</td><td>3</td><td colspan="2"></td></tr>
<tr><td colspan="8">Step 2a: Join L_1 to itself and use the Apriori property to generate candidate 2-itemsets C_2
Step 2b: Scan database <i>D</i> for count of each C_2
Step 2c: Compare C_2 itemsets with minimum support & generate L_2</td></tr>
<tr><td rowspan="7">Join(L_1 L_1) & apply Apriori</td><td>Itemset</td><td rowspan="7">Scan <i>D</i> to get candidate 2-itemset counts</td><td>C_2 Itemset</td><td>Support count</td><td rowspan="7">Compare C_2 support count with minimum support</td><td>L_2 Itemset</td><td>Support count</td></tr>
<tr><td>{A,B}</td><td>{A,B}</td><td>5</td><td>{A,B}</td><td>5</td></tr>
<tr><td>{A,C}</td><td>{A,C}</td><td>4</td><td>{A,C}</td><td>4</td></tr>
<tr><td>{A,E}</td><td>{A,E}</td><td>3</td><td>{A,E}</td><td>3</td></tr>
<tr><td>{B,C}</td><td>{B,C}</td><td>5</td><td>{B,C}</td><td>5</td></tr>
<tr><td>{B,E}</td><td>{B,E}</td><td>3</td><td>{B,E}</td><td>3</td></tr>
<tr><td>{C,E}</td><td>{C,E}</td><td>2</td><td colspan="2"></td></tr>
<tr><td colspan="8">Step 3a: Join L_2 to itself and use the Apriori property to generate candidate 2-itemsets C_3
Step 3b: Scan database <i>D</i> for count of each C_3
Step 3c: Compare C_3 itemsets with minimum support & generate L_3</td></tr>
<tr><td rowspan="3">Join(L_2 L_2) & apply Apriori</td><td>Itemset</td><td rowspan="3">Scan <i>D</i> to get candidate 3-itemset counts</td><td>C_3 Itemset</td><td>Support count</td><td rowspan="3">Compare C_3 support count with minimum support</td><td>L_3 Itemset</td><td>Support count</td></tr>
<tr><td>{A,B,C}</td><td>{A,B,C}</td><td>3</td><td>{A,B,C}</td><td>3</td></tr>
<tr><td>{A,B,E}</td><td>{A,B,E}</td><td>3</td><td>{A,B,E}</td><td>3</td></tr>
</table>

Figure 23. Applying the classical apriori algorithm to determine frequent itemsets of the database in Figure 22.

Consider in greater detail, the steps involved in the determinations of candidate itemsets and the support counts of the candidate itemsets (i.e. the join and scan steps of the algorithm). Consider for example, the join between L_2 and itself, which generates an initial candidate 3-itemsets $C_3^{\prime}$.

1. Join L_2 to itself, where L_2 = {{AB}{AC}{AE}{BC}{BE}}
   - Initial candidate *3-itemsets* $C_3^{\prime}$: {{ABC}{ABE}{ACE}{BCE}}
2. Determinine all the *2-itemset* subsequences of $C_3^{\prime}$:
   - (ABC): {AB}{AC}{BC}

- (ABE): {AB}{AE}{BE}
- (ACE): {AC}{AE}{~~CE~~}
- (BCE): {BC}{BE}{~~CE~~}

3. Use the apriori property to reject all members of C_3' that have one or more *2-itemset* subsequences that are not present in L_2 (i.e. in L_{k-1}), i.e., whose support counts are less than the minimum support count. This leaves candidate *3-itemset* subsequences C_3 = {{ABC}{ABE}}. In the example, CE does not have sufficient support and is crossed out.
4. Determine if the itemsets in C_3 are frequent by scanning the database for all *3-itemset* subsequences and counting the number of occurrences of each C_3 itemsets.

- {ABC}: T001, T006, T007 (3 occurrences)
- {ABE}: T001, T007, T008 (3 occurrences)

We notice that both of the candidate *3-itemsets* selected above meet the minimum support count of 3 in this example, and so they are both included in L_3.

The classical apriori algorithm is summarized in Figure 24.

```
Method
$L_1$ = find_frequent_1-itemsets($D$);
for (k = 2; $L_{k-1} \neq \emptyset$; k++) {
    $C_k$ = **classical_apriori**($L_{k-1}$, *min_sup*);
        for each transaction $t \in D$ { //scan $D$ for counts
            $C_t$ = subset($C_k$, t); //get the subsets of $t$ that are candidates
            for each candidate $c \in C_t$
                c.count++;
        }
    $L_k = \{c \in C_k \mid c.count \geq min\_sup\}$
}
return $L = \cup_k L_k$

procedure classical_apriori($L_{k-1}$:frequent (k-1)-itemsets; *min_sup*: minimum support threshold)
for each iteset $l_1 \in L_{k-1}$
    for each iteset $l_2 \in L_{k-1}$
        if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k-3] = l_2[k-3]) \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ then {
            c = join($l_1$, $l_2$) //join step; generate candidates
            if **has_infrequent_subset**(c,$L_{k-1}$) then
                delete c; //prune step; remove unfruitful candidate
            else add c to $C_k$
        }
return $C_k$

procedure has_infrequent_subset(c: candidate k-itemset,$L_{k-1}$: frequent (k-1)-itemsets)
    //use prior knowledge
for each (k-1)-subsets s of c
    if $s \notin L_{k-1}$ then
        return TRUE
return FALSE
```

Figure 24. The classical Apriori Algorithm. Source: Han and Kamber (2001), with minor rewording.

4.3.2. The Joinless Apriori Algorithm

The main benefit of the apriori property is the reduction in the number of candidate itemsets generated, which leads to a reduction in both the space- and time-complexities of the algorithm. But with the implementation in the classical apriori algorithm, the number of candidate itemsets generated may still be non-trivial, too big to fit into main memory. This, together with the potentially high latency, motivated the development of the joinless apriori algorithm.

The key differences between the joinless and the classical apriori algorithms relate to the stage at which the apriori property is applied as the transactions database is scanned. In *k*th iteration of the classical algorithm, the apriori property is applied to the results of a join of frequent (*k*-1)-itemsets to determine candidate *k*-itemsets, and the database scanned to determine which of the candidate itemsets are frequent. In the joinless algorithm, the join step is eliminated: during the *k*th database scan the apriori property is applied to every transaction of length l ($l \geq k$) to determine candidate *k*-itemsets, and their support counts incremented to enable determination of frequent *k*-itemsets at the end of the iteration. This difference is illustrated below, by repeating Steps 1– 4 of the classical apriori algorithm on Page 72, this time for the joinless algorithm.

1. Scan the transactions database for transactions involving 3 or more items, and determine an initial candidate 3-itemsets $C_3^{\prime}$.

    - T001 (ABCE):{ABC}{ABE} {ACE} {BCE}
    - T003 (ABD): {ABD}
    - T006 (ABC): {ABC}
    - T007 (ABCE):{ABC}{ABE} {ACE}{BCE}
    - T008 (ABE): {ABE}

2. Determine all the *2-itemset* subsequences of the $C_3^{\prime}$

    - T001 (A,B,C): {AB}{AC}{BC}
    - T001 (A,B,E): {AB}{AE}{BE}
    - T001 (A,C,E): {AC}{AE}{~~CE~~}
    - T001 (B,C,E): {BC}{BE}{~~CE~~}
    - T003 (A,B,D): {AB}{AD}{~~BD~~}
    - T006 (A,B,C): {AB}{AC}{BC}
    - T007 (A,B,C): {AB}{AC}{BC}
    - T007 (A,B,E): {AB}{AE}{BE}
    - T007 (A,C,E): {AC}{AE}{~~CE~~}
    - T007 (B,C,E): {BC}{BE}{~~CE~~}
    - T008 (A,B,E): {AB}{AE}{BE}

3. Use the apriori property to reject all *3-itemsets* with one or more *2-itemset* subsequences that do not appear in L_2 (i.e. L_{k-1}), i.e., whose support counts are less than the minimum support count (recall L_2 = {{AB}{AC}{AE}{BC}{BE}}). In the example, CE and BD do not have support and are crossed out.
4. Count the occurrences of each *3-itemset* subsequences that are not rejected in the previous step to get their support counts. In this example, both {ABC} and {ABE} meet minimum the support count of 3, and so are added to L_3.

We note that the absence of the join step results in general, to better space complexity in the joinless apriori algorithm. This is so because the apriori property is applied to individual transactions, which usually involve much fewer items than the total number of items in the transactions database. Figure 25 summarizes the steps involved in finding frequent itemsets for the database of Figure 22, using the joinless apriori algorithm. As can be seen from the figure, exactly the same frequent itemsets are determined by both the classical and joinless Apriori algorithms. Figure 26 shows the joinless apriori algorithm.

Step 1a: Scan database *D* for count of each candidate 1-itemset C_1
Step 1b: Compare C_1 itemsets with minimum support & generate L_1

		C_1 Itemset	Support count	Compare C_1 support count with minimum support	L_1 Itemset	Support count
	Scan *D* to get candidate 1-itemset counts	{A}	6		{A}	6
		{B}	7		{B}	7
		{C}	6		{C}	6
		{D}	1		{E}	3
		{E}	3			

Step 2a: Scan database *D* for transactions >= 2
Step 2b: Determine 2-itemset subsequences and apply Apriori property to get C_2
Step 2c: Count each subsequence to determine support count for C_2
Step 2d: Compare C_2 itemsets with minimum support & generate L_2

	Transaction		C_2 Itemsets	C_2 Itemset	Support count		L_2 Itemset	Support count
Scan *D* for transactions with 2 or more items	{A,B,C,E}	Candidate 2-itemset = 2-itemset subsequences + Apriori	{A,B} {A,C} {A,E} {B,C} {B,E} {C,E}	{A,B}	5	Compare C_2 support count with minimum support	{A,B}	5
	{B,C}		{B,C}	{A,C}	4		{A,C}	4
	{A,B,D}		{A,B}	{A,E}	3		{A,E}	3
	{A,C}		{A,C}	{B,C}	5		{B,C}	5
	{B,C}		{B,C}	{B,E}	3		{B,E}	3
	{A,B,C}		{A,B} {A,C} {B,C}	{C,E}	2			
	{A,B,C,E}		{A,B} {A,C} {A,E} {B,C} {B,E} {C,E}					
	{A,B,E}		{A,B} {A,E} {B,E}					

Step 3a: Scan database *D* for transactions >= 3
Step 3b: Determine 3-itemset subsequences and apply Apriori property to get C_3
Step 3c: Count each subsequence to determine support count for C_3
Step 3d: Compare C_3 itemsets with minimum support & generate L_3

	Transaction		C_3 Itemsets	C_3 Itemset	Support count		L_3 Itemset	Support count
Scan *D* for transactions with 3 or more items	{A,B,C,E}	Candidate 3-itemset = 3-itemset subsequences + Apriori	{A,B,C} {A,B,E}	{A,B,C}	3	Compare C_3 support count with minimum support	{A,B,C}	3
	{A,B,D}			{A,B,E}	3		{A,B,E}	3
	{A,B,C}		{A,B,C}					
	{A,B,C,E}		{A,B,C} {A,B,E}					
	{A,B,E}		{A,B,E}					

Figure 25. Applying the joinless apriori algorithm to determine frequent itemsets for the database in Figure 22.

```
**Method**
$L_1$ = find_frequent_1-itemsets(*D*);
for (k = 2; $L_{k-1} \neq \varnothing$; k++) {
    for each transaction $t \in D$ {
        $C_t = t \mid card(t) \geq k$ {//scan *D* for all transactions with itemset size $\geq$ k
            $C_t$ = subset($C_k$, t); //get the subsets of *t* that are candidates
            c += subseq($C_t$, k); //add k-subsequences of transaction *t* to list of candidate itemsets
    }
    $C_k$ = **joinless_apriori**($L_{k-1}$,$C_t$, *min_sup*);
    $L_k = \{c \in C_k \mid c.count \geq min\_sup\}$
}
return $L = \cup_k L_k$

**procedure joinless _apriori**($L_{k-1}$:frequent (k-1)-itemsets; c: list of k-itemsets in *D min_sup*: min. support threshold)
        for each itemset $i \in c$ {
    i.count++;
    if **has_infrequent_subset**(i,$L_{k-1}$) then
        delete c; //prune step; remove unfruitful candidate
    else add c to $C_k$;
}
return $C_k$

**procedure has_infrequent_subset**(c: candidate k-itemset,$L_{k-1}$: frequent (k-1)-itemsets)
    //use prior knowledge
for each (k-1)-subsets s of c
    if $s \notin L_{k-1}$ then
        return TRUE
return FALSE
```

Figure 26. The joinless apriori algorithm.

4.3.3. C-Annotated Databases and the Joinless Apriori Algorithm

The structure of the transactions data used in this study is different from the general format used for example, in market basket analysis. In the later case, the frequent itemsets that are mined shoe the correlation relations among all the items in each transaction of the database. In the former case however, the data in each transaction comprises one or more navigation pages, followed by a content page, with the navigation page(s) representing the association rule antecedent, and the content page, the rule consequent. In this study, we refer to such transactions databases with a well known consequent for every row as *consequent-annotated*, or *c-annotated*.

To illustrate the difference between mining a consequent-annotated transactions database, and a general transactions database, consider two examples involving market basket analysis. Assume that milk, bread, and cereals are the entries in the transactions database. The goal of association rule mining in the general case would be to discover strong association rules of the following forms:

$$buys(X, \text{“milk”}) \wedge buys(X, \text{“bread”}) \Rightarrow buys(X, \text{“cereals”})$$

buys(X, “milk”) $\wedge$ buys(X, “cereals”) $\Rightarrow$ buys(X, “bread”)

buys(X, “cereals”) $\wedge$ buys(X, “bread”) $\Rightarrow$ buys(X, “milk”)

buys(X, “cereals”) $\Rightarrow$ buys(X, “milk”) $\wedge$ buys(X, “bread”)

buys(X, “bread”) $\Rightarrow$ buys(X, “milk”) $\wedge$ buys(X, “cereals”)

buys(X, “milk”) $\Rightarrow$ buys(X, “cereals”) $\wedge$ buys(X, “bread”)

buys(X, “milk”) $\Rightarrow$ buys(X, “cereals”)

buys(X, “milk”) $\Rightarrow$ buys(X, “bread”)

buys(X, “bread”) $\Rightarrow$ buys(X, “cereals”)

buys(X, “bread”) $\Rightarrow$ buys(X, “milk”)

buys(X, “cereals”) $\Rightarrow$ buys(X, “bread”)

buys(X, “cereals”) $\Rightarrow$ buys(X, “milk”)

On the other hand, if the database is consequent-annotated, for example with cereals as the item of interest, then the only association rules that need be generated should be of the forms:

buys(X, “milk”) $\wedge$ buys(X, “bread”) $\Rightarrow$ buys(X, “cereals”)

buys(X, “milk”) $\Rightarrow$ buys(X, “cereals”)

buys(X, “bread”) $\Rightarrow$ buys(X, “cereals”)

Clearly, because of the constraint of limiting the association rules to a predetermined consequent, the search space required for the determination of association rules is much smaller in the case of a consequent-annotated database than in the general case. The procedure to obtain frequent itemsets for a c-annotated transactions database is identical to the case for a general transactions database, except that: (1) one of the items in each transaction is annotated as the consequent of the association rule for that transaction and all of its subsequences; and (2) candidate frequent itemsets and frequent itemsets are generated only for the unannotated items of the transactions. For example, if the last item of each transaction in the database in Figure 22 is annotated as the consequent, the resulting transactions database is given in Figure 27 and the procedure to obtain frequent itemsets using the joinless apriori algorithm is summarized in Figure 28.

TID	Transaction Items	Rule Consequent
T001	A,B,C	E
T002	B	C
T003	A,B	D
T004	A	C
T005	B	C
T006	A,B	C
T007	A,B,C	E
T008	A,B	E

Figure 27. Sample consequent-annotated database used to illustrate the application of the joinless apriori algorithm to a c-annotated database.

<table>
<tr><td colspan="12">Step 1a: Scan database <i>D</i> transactions for count of each candidate 1-itemset C_1
Step 1b: Compare C_1 itemsets with minimum support & generate L_1</td></tr>
<tr><td colspan="2" rowspan="8"></td><td colspan="3" rowspan="8">Scan <i>D</i> to get candidate 1-itemset counts</td><td>C_1 Itemset</td><td>Conse-quent</td><td>Support count</td><td rowspan="8">Compare C_1 support count with minimum support</td><td>L_1 Itemset</td><td>Conse-quent</td><td>Support count</td></tr>
<tr><td>{A}</td><td>C</td><td>2</td><td>{B}</td><td>C</td><td>3</td></tr>
<tr><td>{B}</td><td>C</td><td>3</td><td>{A}</td><td>E</td><td>3</td></tr>
<tr><td>{A}</td><td>D</td><td>1</td><td>{B}</td><td>E</td><td>3</td></tr>
<tr><td>{B}</td><td>D</td><td>1</td><td></td><td></td><td></td></tr>
<tr><td>{A}</td><td>E</td><td>3</td><td></td><td></td><td></td></tr>
<tr><td>{B}</td><td>E</td><td>3</td><td></td><td></td><td></td></tr>
<tr><td>{C}</td><td>E</td><td>2</td><td></td><td></td><td></td></tr>
<tr><td colspan="12">Step 2a: Scan database <i>D</i> transactions for transactions >= 2
Step 2b: Determine 2-itemset subsequences and apply Apriori property to get C_2
Step 2c: Count each subsequence to determine support count for C_2
Step 2d: Compare C_2 itemsets with minimum support & generate L_2</td></tr>
<tr><td rowspan="6">Scan <i>D</i> for transactions with 2 or more items</td><td>Transaction</td><td rowspan="6">Candidate 2-itemset = 2-itemset subsequences + Apriori</td><td>C_2 Itemsets</td><td>Conse quent</td><td>C_2 Itemset</td><td>Conse-quent</td><td>Support count</td><td rowspan="6">Compare C_2 support count with minimum support</td><td>L_2 Itemset</td><td>Conse-quent</td><td>Support count</td></tr>
<tr><td>{A,B,C}</td><td>{A,B}{A,C}{B,C}</td><td>E</td><td>{A,B}</td><td>E</td><td>3</td><td>{A,B}</td><td>E</td><td>3</td></tr>
<tr><td>{A,B}</td><td>{A,B}</td><td>D</td><td>{A,C}</td><td>E</td><td>2</td><td></td><td></td><td></td></tr>
<tr><td>{A,B}</td><td>{A,B}</td><td>C</td><td>{B,C}</td><td>E</td><td>2</td><td></td><td></td><td></td></tr>
<tr><td>{A,B,C}</td><td>{A,B}{A,C}{B,C}</td><td>E</td><td>{A,B}</td><td>D</td><td>1</td><td></td><td></td><td></td></tr>
<tr><td>{A,B}</td><td>{A,B}</td><td>E</td><td>{A,B}</td><td>C</td><td>1</td><td></td><td></td><td></td></tr>
</table>

Figure 28. Applying the joinless apriori algorithm to determine frequent itemsets for the c-annotated database in Figure 27.

4.3.3.1. Generation of Association Rules from Frequent Itemsets

Frequent itemsets correspond to potential association rules: they may be pruned if they do not meet a minimum support or rule confidence. The support of a rule is the fraction of items in the rule that occur together in the set of transactions, and can be computed using the following formula:

$$Support(A \Rightarrow B) = P(A \cup B) = |(A \cup B)| \,/\, |transactions\ database| \qquad (3)$$

Confidence is the percentage of the rule antecedent that leads to the consequent and is computed as the conditional probability of the rule consequent, given the antecedent:

$$confidence(A \Rightarrow B) = P(B/A) = support\_count(A \cup B)/support\_count(A) \qquad (4)$$

For example, for the c-annotated database of Figure 27, the following association rules can be derived from frequent itemsets L_1 and L_2 in Figure 28; rules that meet both a predetermined minimum support and confidence score are kept, and the rest discarded.

$B \Rightarrow C$ [support = 3/8; confidence = 3/7]

$A \Rightarrow E$ [support = 3/8; confidence = 3/6]

$B \Rightarrow E$ [support = 3/8; confidence = 3/7]

$A \wedge B \Rightarrow E$ [support = 3/8; confidence = 3/5]

4.3.3.2. Implementation of the Classical and Joinless Apriori Algorithms on a c-annotated Database

Figure 29 shows the data structure used in this study to mine a c-annotated database for association rules. The transactions database contained user navigation paths with the following structure for each record: {url_1, url_2, …, url_n, url_I: $n \geq 1$) where, $\text{url}_1 \ldots \text{url}_n$ were the navigation urls that corresponded to a page of interest, ulr_I. The objective of the mining process was to generate rules of the form:

$$user\_follows(X, \text{“}url_i\text{”}) \wedge user\_follows(X, \text{“}url_j\text{”}) \wedge \ldots \wedge user\_follows(X, \text{“}url_n\text{”}) \Rightarrow user\_interested\_in(X, \text{“}url_I\text{”})$$

The data structure comprises an interconnection of linked lists. The backbone of the structure (top of figure) comprised *w* hash tables (*w* = number of distinct association rule antecedent lengths), each with *h* buckets, where *h*, the hash table size was set to 251. The first hash table dealt with all association rules comprising 1 navigation page and one content page, the second hash table dealt with association rules comprising 2 navigation pages and one content page, etc. For a given entry in the transactions database with antecedent length *l*, the following hash function was used to determine which hash bucket in hash table *l* would be used to handle the transaction:

$$hash\_bucket = \left(\sum navigation\ \&\ content\ \ page\ IDs\right) \bmod h \qquad (5)$$

For a given navigation pattern, the hash tables reduced the search space required to locate the pattern by directing the search first to the hash table with the corresponding length, and then, to the bucket that corresponded to that pattern's hash value.

Each hash bucket had a pointer to a linked list (middle of figure) of all the navigation pages that hashed to that bucket. A count (pattern_count) was maintained for the number of occurrences of each of these navigation patterns. This count was useful in determining the confidence values of association rules that had the corresponding navigation pages at the left hand side of the association rule.

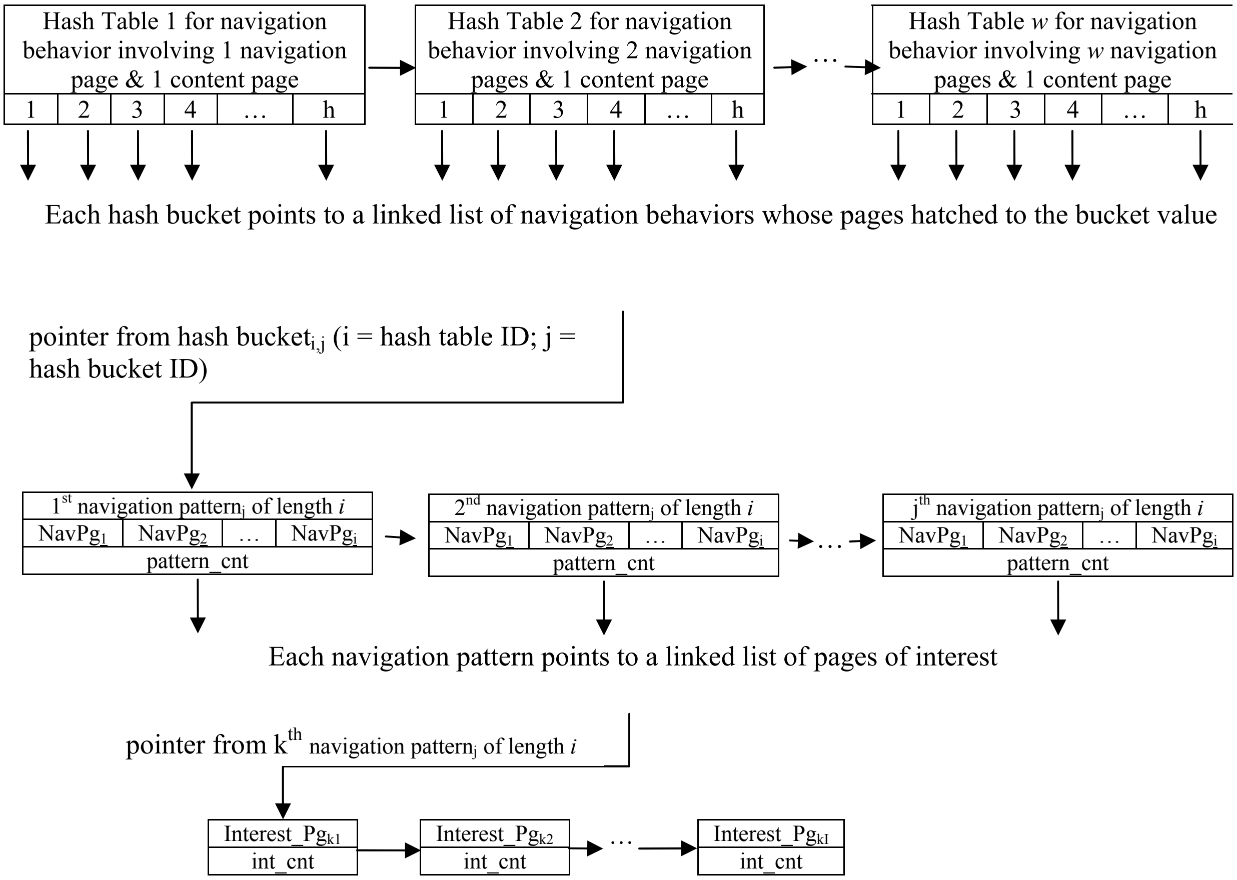

Hash Table 1 for navigation behavior involving 1 navigation page & 1 content page
1
2
3
4
…
h
Hash Table 2 for navigation behavior involving 2 navigation pages & 1 content page
1
2
3
4
…
h
…
Hash Table w for navigation behavior involving w navigation pages & 1 content page
1
2
3
4
…
h
Each hash bucket points to a linked list of navigation behaviors whose pages hatched to the bucket value
pointer from hash bucket$_{i,j}$ (i = hash table ID; j = hash bucket ID)
1st navigation pattern$_j$ of length *i*
NavPg$_1$
NavPg$_2$
…
NavPg$_i$
pattern_cnt
2nd navigation pattern$_j$ of length *i*
NavPg$_1$
NavPg$_2$
…
NavPg$_i$
pattern_cnt
…
jth navigation pattern$_j$ of length *i*
NavPg$_1$
NavPg$_2$
…
NavPg$_i$
pattern_cnt
Each navigation pattern points to a linked list of pages of interest
pointer from kth navigation pattern$_j$ of length *i*
Interest_Pg$_{k1}$
int_cnt
Interest_Pg$_{k2}$
int_cnt
…
Interest_Pg$_{kl}$
int_cnt

Figure 29. Data structure used to determine frequent itemsets from a transactions database.

Finally, because the same navigation pages corresponded to different interest pages, a pointer was used to link every navigational pattern to a linked list of the various pages of interest that were observed (bottom of figure). A count (int_cnt) was maintained for the number of occurrences of these pages of interest for each set of navigation pages, and was used to compute the support of the rule.

4.3.4. Experiment 1: Evaluation of Joinless Apriori Algorithm

The new joinless apriori algorithm proposed in this study was evaluated relative to the classical apriori algorithm by running simulations of the two algorithms for varying values set for the following parameters: minimum support count (which determines total number of rules generated), and mean transaction record length, to determine which, if any, of the algorithms works better under these varying conditions. Performance statistics relating to execution time, association rule length, and association rule

Minimum support count	Maximum acceptable transaction antecedent length L_{max} (Determines mean association rule length)					
	5	6	7	8	9	10
1	Performance statistics for classical and joinless apriori algorithms					
2						
3						
4						
5						
6						
7						

Figure 30. Experimental design for the evlauation of the joinless apriori algorithm.

count were then collected, and compared. All the simulations were run on a SunBlade 2500 workstation running the Sun Solaris 9 UNIX operating system. The programs were all written in ANSI C, and compiled using the C compiler that comes with the operating system.

Data and Procedure. The data comprised the user access log for the website of SIS for the months of June to August 2004. The raw data were stored in the common log format (W3C, 2003) and totaled about 500MB. We followed the procedure described in Section 4.2 to generate the transactions database, and then ran both the classical and joinless apriori algorithms on the transactions database for the following values of minimum support count: 1, 2, 12, 24, 59, 118, and 235, (corresponding to the following fractions of the transaction database size: 0.000005, 0.00001, 0.00005, 0.0001, 0.00025, 0.0005, 0.001), and for different average association rule lengths. Average association rule lengths were controlled by varying the maximum acceptable transaction length L_{max} between 5 and 10. For each run, all transactions whose lengths were larger than L_{max} were ignored. The experimental design used is summarized in Figure 30.

4.4. Design of the CLARANS Clustering Algorithm for High Dimensional Data

As discussed in Section 3.2.4, the CLARANS clustering algorithm was investigated to determine its suitability in high dimensional spaces comprising large document collections, since personalization systems operate in such environments. We start by discussing the memory/computation time tradeoff that was made because of the large size and dimensionality of the data sets considered (Section 4.4.1). This is followed by a review of the PAM and CLARA clustering algorithms on which CLARANS is based, and of CLARANS (Section 4.4.2), choice of similarity measure (Section 4.4.3), and the evaluation procedure (Section 4.4.4).

4.4.1. Memory/Computation Time Tradeoffs in CLARANS

In a main-memory clustering algorithm like CLARANS, it is desirable to process all or most of the computations using data held in primary memory, and make as few accesses as possible to secondary storage. This is not much of a problem for low dimensional data, where relatively few bytes are required to represent data objects. And with such few dimensions, both the computation of object-object similarities and the clustering algorithm can be done efficiently in main memory.

For very high dimensional data however, main memory is a much more limited resource as it may not be feasible to store all the data objects and their object-object similarity scores in main memory at the same time. Computation time also becomes a critical factor, as it takes much longer to compare the numerous dimensions for similarity computations.

The memory/computation time tradeoff adopted here was to divide the clustering process into off- and on-line stages. The off-line stage computed object-object similarity scores by loading the object vectors in turn from secondary storage into main memory, and computing their similarity scores. The many accesses to the relatively slow, secondary memory made the process of computing similarities slow. The on-line process involved the discovery of clusters, and was possible because all that needed to be loaded into main memory were the similarity scores and the clustering algorithm.

4.4.2. Review of PAM, CLARA and CLARANS

PAM (Kaufman & Rousseeuw, 1990) is a partitioning algorithm that generates *k* clusters by finding *k* representative objects, or *medoids*, and assigning each of the remaining objects to the medoid that is most similar to it. The *quality* of a clustering is defined as the average dissimilarity between the objects in each cluster and their associated medoid (the lower the average dissimilarity, the better the clustering). The algorithm begins with *k* arbitrary medoids and their associated clusterings. Then, for each medoid object, the cost (i.e. change in average dissimilarity) is computed if the medoid were replaced with a non-medoid object, and new clusterings formed. If the cost is negative, meaning there is an improvement in the quality of the clustering, the medoid is replaced by the non-medoid, and the process continues until the quality of the clusters cannot be improved further. Kaufman and Rousseeuw (1990) and Ng and Han (1994; 2002) provided complete details on the PAM algorithm.

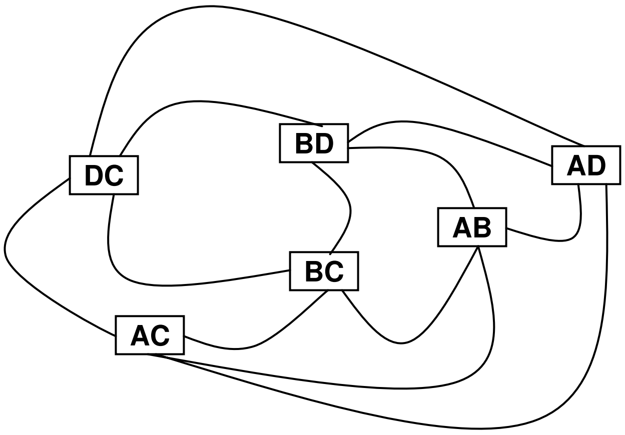


Figure 31. Graphical representation of the search space for the PAM algorithm. This example illustrates a search space with 4 objects (A,B,C,D, i.e., n = 4), and with 2 clusters (k = 2). Each node has k(n-k) neighbors.

Conceptually, PAM can be viewed as a search for a minimum node (the node that corresponds to the best quality of clustering) on a graph $G_{n,k}$, where n is the number of objects in the set being considered, and k is the number of clusters to be formed. Such a graph has the following properties: (1) each node on the graph comprises a set of k objects $\{O_{m1},\ldots,O_{mk}\}$ corresponding to medoids; (2) two nodes in the graph, S_1 and S_2 are neighbors and linked with an arc if they differ in only one object, i.e., if $\left|S_1 \quad S_2\right| \quad k \quad 1$. The graph has *k(n-k)* neighbors, as illustrated in the simple example in Figure 31, which shows the graph $G_{4,2}$. Moving from one node containing object O_m to a neighboring node containing object O_p if the two nodes differ in O_p and O_m (i.e. $O_m \in S_1$ and $O_p \in S_2$, but $O_m, O_p \notin S_1 \cap S_2$), is equivalent to replacing medoid O_m with non-medoid O_p. The problem with PAM is that it is computationally too expensive for large values of n and k; a single iteration is of $O(k(n-k)^2)$ (Ng & Han, 1994, 2002).

CLARA (Kaufman & Rousseeuw, 1990) was designed to handle large datasets. CLARA is similar to PAM in that it involves a search for a minimum node in a graph structure. But instead of finding medoids from the whole dataset, CLARA relies on a random sampling approach. Several samples are drawn from the data set, and the PAM algorithm applied to each sample. The quality of the clustering, based on the entire dataset, is obtained for each sample, and the best quality medoids considered the best for the entire data set. CLARA is thus equivalent to searching for the best node in a subgraph $G'_{n,k}$ of graph $G_{n,k}$. The problem with CLARA is that if the minimum node is not in the selected sample, it will never be found. A complete description of CLARA is provided by Kaufman and Rousseeuw (1990).

The objective of CLARANS is to overcome this shortcoming of CLARA. CLARANS takes two parameters (*numlocal* and *maxneighbors*). Parameter *numlocal* determines how many times the algorithm will iterate to find the best quality medoids, i.e., how many local minima will be determined so that they can be compared, and the one with the best quality selected. The second parameter *maxneighbors* determines for each node, how many neighboring nodes should be searched for a node with a lower cost before the algorithm concludes that the current node is a local minimum. The expectation is that with graph $G_{n,k}$ being as highly connected as it is, searching a sample of neighboring nodes should usually find a lower cost node if one exists. And even if the current local minimum is not the global minimum, repeating the search *numlocal* times should lead to very good clustering.

CLARANS is similar to PAM in that the search in not confined to a subgraph as in CLARA; CLARANS (like CLARA) is different from PAM however, in that, not all the nodes of a given graph are searched. CLARANS is also different from CLARA in that instead of confining the search to a set of nodes, determined at the start of the algorithm by a sample of objects, CLARANS selects a sample of neighboring nodes in each step of the search. Complete details on algorithm CLARANS were presented by Ng and Han (1994; 2002)

4.4.3. Similarity Measure

In this section, we present properties of any dissimilarity metric that are useful for cluster analysis, and observe that Euclidean distances are not suitable in clustering high dimensional data, making it necessary to use a non-Euclidean dissimilarity metric. The properties are listed below.

Symmetry. This property states that given two objects *a* and *b*, the distance or dissimilarity, *d*, between them satisfies the expression $d(a,b) \quad d(b,a) \quad 0.$

Triangular Inequality. This property states that the dissimilarities between three objects *a*, *b*, and *c* satisfy the expression $d(a,b) \quad d(a,c) \quad d(b,c).$

Distinguishability of Non-identicals. If the dissimilarity between two objects *a* and *b* is not zero, then the objects are different, i.e., $\text{if } d(a,b) \quad 0, \text{then a} \quad \text{b.}$

Indistinguishability of Identicals: If two objects *a* and *b* are identical, then the dissimilarity between them is zero, i.e., $\text{if a} = \text{b, then } d(a,b) = 0$

A commonly used measure for dissimilarity in clustering algorithms is the Euclidean distance. This is fine in 2-D space, but in high dimensional space, the Euclidean distance does not work well as (1) in high dimensional space, distances between vectors are not very distinguishable from each other and are almost meaningless (the Euclidean distances from any two points tend to be very similar), and (2) data in high-dimensional spaces tend to be sparse; Euclidean distance does not take this into consideration. Several other similarity measures like the cosine measure have been found to be useful in IR and related fields, if the documents are represented as vectors whose components are tokens (typically words) derived from the document. One advantage in using the cosine measure is the fact that the computation treats small documents (as it does larger documents) *fairly* in the sense that the measure is normalized for document length. The inverse of the cosine similarity, or angle, between two vectors *P* and *Q* (*arcos*(*sim*(*P,Q*))), where *sim*(*P,Q*) is the cosine similarity between the vectors, meets the conditions listed above for a suitable dissimilarity metric for cluster analysis. The metric has been used by a number of researchers (e.g., Ullman (2003b) and Yongling (2004)), and was adopted for use in this study. It should be noted that even with a change of dissimilarity metric, other problems persist in high dimensional data analysis. For example, it is likely that two vectors that are very similar in most dimensions may be very dissimilar in one or more other dimensions, making them appear very dissimilar (Ullman, 2003a, 2003c).

4.4.4. Evaluation of CLARANS

As mentioned in Section 4.4.2, the CLARANS clustering algorithm requires 2 parameters: *numlocal* which determines the number of times the algorithm should iterate to find the best quality medoids, and *maxneighbors* which determines for each node of the graph representation of the clustering problem, the optimal number of neighboring nodes to search for a node with a lower cost before the algorithm concludes that the current node is a local minimum. The CLARANS algorithm was evaluated in 2 stages: the first was to determine optimal values for parameters *numlocal* and *maxneighbors*, and the second to evaluate the performance (effectiveness and efficiency) of CLARANS with respect to the PAM and CLARA algorithms on which it is based.

Determination of optimal values of parameters *numlocal* and *maxneighbors* involved running a batch of simulations of the algorithm with these parameters set to different values to determine which one resulted in the best quality clusters of high dimensional data objects. Another set of simulations was run to compare the efficiency (execution times) and quality of clusters discovered by the CLARANS, PAM and CLARA algorithms. All the simulations were performed on a SunBlade 2500 workstation running the Sun Solaris 9.1 UNIX operating system. The programs were all written in ANSI C, and compiled using the *C* compiler that comes with the operating system.

Test Data. We simulated two high dimensional object spaces by generating sets of vectors (representing objects to be clustered) with dimensionalities of *d1* and *d2* to represent documents in *d1*- and *d2*-dimensional spaces. The generated vectors were constrained to belong to predetermined clusters, and it was the task of the 3 clustering algorithms being compared to attempt to assign the vectors to their correct clusters. 16 sets of simulated documents divided into two groups of 8 sets each were generated. The first group comprised document collections of sizes 1100 in a 4500-dimensional space, and sizes 4500 in 3000-dimensional space. There were 4 sets of 1100-sized documents comprising 10, 20, 30, and 40 clusters respectively, and 4 sets of 4500-sized documents, respectively comprising 15, 30, 45, and 60 clusters. The second group of 8 simulated document sets comprised the same object and cluster sizes as the first. The difference between the two groups was in their inter-cluster and intra-cluster similarities: the first group was designed to have larger intra-cluster similarities and smaller inter-cluster similarities than the second.

Our choices of document collection sizes and dimensionalities were influenced by two factors: (1) the values of these parameters in the collection of web pages in the web site which were clustered based on the similarities of their contents, and (2) main memory limitations.

- There were 4490 web pages, and 3027 index terms (dimensionality), spread across the web pages. The 4500-sized objects were representative of these web pages.
- We also wanted to investigate the performance of the clustering algorithms in higher dimensional space, and chose a dimensionality of 4500. Because of main memory limitations, we restricted the number of documents with this dimensionality to 1100.

We used the notation $R_{n,k}$ to represent a dataset comprising n documents and k clusters, and the notation $R'_{n,k}$ and $R''_{n,k}$ to specify the first and second groups respectively. We next consider how the clusters were formed.

Figure 32 is a 2-D illustration of the approach used to generate documents used in the evaluation, except that we worked with 3000- and 4500-, and not 2-dimensional objects as the figure suggests. Each square in the figure has length b, and delineates a bounded region that holds objects in a cluster. By randomly generating vectors with components of length l_d in the *dth* dimension ($0 < l_d \leq b$) of the document space, we can create a cluster bounded by the square near the origin. A translation operation can then be applied to the cluster to place it in a different area of the 2-D space. The process of generating a cluster near the origin and applying a different translation operator can be repeated until a sufficient number of document clusters are generated.

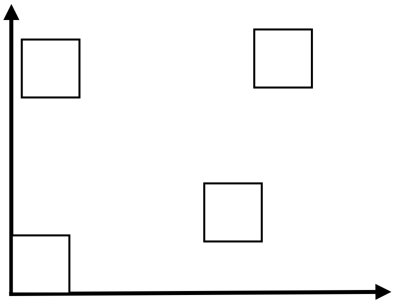


Figure 32. Illustrating generation of test data used to evaluate CLARANS, CLARA, and PAM.

Table 1. *Intra- and inter-cluster cosine similarity scores for generated data.*

	$R'_{n,k}$		$R''_{n,k}$	
Dataset	Intracluster cosine similarity	Intercluster cosine similarity	Intracluster cosine similarity	Intercluster cosine similarity
$R_{1100,10}$	0.90-0.92	0.46-0.51	0.82-0.86	0.63-0.68
$R_{1100,20}$	0.82-0.88	0.52-0.58	0.79-0.82	0.67-0.72
$R_{1100,30}$	0.83-0.85	0.57-0.62	0.78-0.81	0.68-0.73
$R_{1100,40}$	081.-0.84	0.59-0.65	0.77-0.80	0.69-0.74
$R_{4500,15}$	0.87-0.90	0.49-0.56	0.80-0.84	0.64-0.71
$R_{4500,30}$	0.83-0.86	0.55-0.63	0.77-0.81	0.67-0.74
$R_{4500,45}$	0.80-0.84	0.59-0.67	0.76-0.80	0.69-0.75
$R_{4500,60}$	0.79-0.83	0.61-0.69	0.76-0.79	0.69-0.75

One difficulty handling objects in such high dimensional spaces is that a single dimension cannot adequately distinguish one object from the next. Our approach to creating clusters for dataset $R_{n,k}$ was first to earmark the first *n/k* documents for the first cluster, the next *n/k* documents for the second cluster, and so on. Second, we earmarked the first *N/k* dimensions to use for translating the first cluster generated near the origin of the document space to a different region of the space, the next *N/k* dimensions for the second cluster, etc., where *N* is the dimensionality of the space. Then we generated each cluster near the origin of the document space bounded by a region not greater than length *b* along each dimension, and performed the translation operation along the designated dimensions by adding a constant separation value *s*, to the designated components for each of the vectors in the cluster. In this study, we set the separation variable *s* to 12 for each of the two groups of data sets that were generated. The value of boundary variable *b* determined the intra-cluster similarities (the smaller the value of b, the more packed together the documents would be, and the greater their intra-cluster similarities). Variable *b* was set to *6* for datasets $R'_{n,k}$, and to 12 for datasets $R''_{n,k}$. Table 1 shows the distribution of intra- and inter-cluster cosine similarity values for the generated datasets, obtained by examining the similarity values of objects within the same clusters and across clusters.

Dissimilarity Matrix. With the simulated document vectors in secondary storage, the object-object cosine similarity values were computed and stored as an upper triangular matrix, to minimize both the time and storage requirements. The steps involved iteratively reading a document vector into primary memory and for every such vector read, reading the document vectors that followed it, and computing the cosine similarity value as explained in Appendix A.

Calibration of CLARANS. As mentioned before, CLARANS takes two parameters, *maxneighbors* and *numlocal*. Ng and Han (2002) suggested that the optimal value for *maxneighbors* is a function of a fraction of number of neighboring nodes *k(n-k)* to each node in the graph representation of the problem. The steps involved in the calibration of the algorithm are detailed in the following paragraphs.

First, we determined what fraction of the neighboring nodes to use for parameter *maxneighbors* that would ensure that the algorithm resulted in high quality clusters without at the same time being too expensive in terms of execution time. That involved running CLARANS with *maxneighbors* set to varying percentages (*p*) of *k(n-k)*, up to a maximum value of 100%, or 10000 neighbors, whichever was smaller, and the *numlocal* parameter set to a high value (10 in this study). The number of neighboring nodes examined was limited to 10000 because of the excessive cost in time that would result otherwise. Three iterations of the algorithm were run, and for each value of *p*, we determined the average cluster dissimilarity and average execution time required to run the algorithm. The value of *p* that resulted in good quality clusters without requiring an excessive amount of processing time was considered optimal.

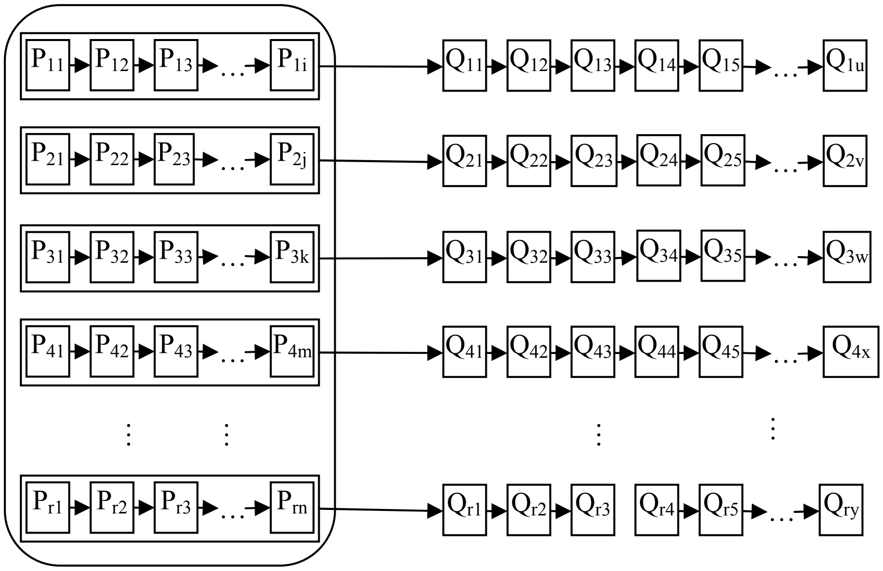
The second stage in the calibration of CLARANS involved determining an optimal value for parameter *numlocal*. This was done by first determining the execution times and average dissimilarity values for values of parameter *numlocal* between 1 and 5 for the optimal value of the *maxneighbors* parameter. As with *maxneighbors*, the value for *numlocal* that corresponded to high quality clusters without a very heavy cost on computation time was chosen to be optimal.

Comparing PAM, CLARA, and CLARANS. Following the calibration of CLARANS, the three clustering algorithms were evaluated for the generated data sets based on their execution times and average intra-cluster dissimilarity values.

4.5. Design of the Collaborative Filtering Prediction Models

The last objective of this work (discussed in Section 3.2.3) was to investigate the design of collaborative filtering prediction models for current user interests as they browse a website. Classical collaborative filtering computations require input data that are equivalent to a matrix, with the rows representing users, and the columns their ratings of objects. The association rules mined as described in Section 4.3.3 were used as collaborative filtering data as outlined below:

- Every unique rule antecedent (i.e., navigation pages) was considered equivalent to a user in classical collaborative filtering (i.e., row label)
- Every rule consequent (i.e. corresponding page of interest) was considered equivalent to an item of interest rated by the user (i.e. column label) in classical collaborative filtering



P_{11} P_{12} P_{13} … P_{1i}
P_{21} P_{22} P_{23} … P_{2j}
P_{31} P_{32} P_{33} … P_{3k}
P_{41} P_{42} P_{43} … P_{4m}
P_{r1} P_{r2} P_{r3} … P_{rn}
Q_{11} Q_{12} Q_{13} Q_{14} Q_{15} … Q_{1u}
Q_{21} Q_{22} Q_{23} Q_{24} Q_{25} … Q_{2v}
Q_{31} Q_{32} Q_{33} Q_{34} Q_{35} … Q_{3w}
Q_{41} Q_{42} Q_{43} Q_{44} Q_{45} … Q_{4x}
Q_{r1} Q_{r2} Q_{r3} Q_{r4} Q_{r5} … Q_{ry}

Figure 33. Adjacency list representation of association rule matrix used in the design of the collaborative filtering-based recommendation models. Each row on the left of the figure shows a unique linked list of web page sequences (corresponding to an association rule antecedent) pointed to by the corresponding array index. Each row on the right of the figure shows a linked list of web pages of interest corresponding to the rule antecedent.

- The rule confidence value for a given rule antecedent and consequent was considered equivalent to the user rating for that rule and page of interest
- For each rule, all pages of interest that the mining process did not find interesting were considered to have a rating of zero

The resulting data were a very large and sparse matrix with each row having only a few page ratings. The data structure shown in Figure 33, also referred to elsewhere in this report as the association rules matrix, was used to hold the matrix data efficiently. The data structure was an array of linked lists, where each array index was a 2-pointer data structure: the first pointer pointed to a list of pages that corresponded to a unique rule antecedent (*user*), and the second, to a list of pages of interest with ratings greater than zero.

Predicting the interests score for web pages for a user with a given navigation behavior involved the following stages:

- Store the user's navigation behavior in a fixed-width window *w* (an integer vector) of size *n*, where the contents of *w* were IDs of the *n* pages last visited by the user
- Compute the *k* nearest neighboring navigation behaviors (neighborhood formation) from the association rules matrix (equivalent to finding the *k* most similar users)
- Predict the confidence values (ratings) for the recommendable hyperlinks for the current user, based on the ratings of the *k* nearest neighbors. For a fully functional recommender system, these predicted confidence scores would be used to compute recommendation scores for the various web pages, which would guide the decision on which pages to recommend to an the user

Computing the k Nearest Neighboring Navigation Behaviors. The following steps were followed to determine which *k* row vectors of the association rules matrix of Figure 33 to use in the prediction of recommendable page ratings for an active user of the system. Recall (from Section 3.2.3) that two neighborhood formation approaches were implemented: center-based (Collaborative Filtering Model I) and aggregate-based (Collaborative Filtering Model II).

In the center-based scheme, the following steps were involved:

1. Compute the cosine similarity between the current user navigation behavior vector (window *w*), and each of the row vectors of the association rules matrix
2. Select the *k* row vectors whose similarities with the current user navigation vector exceed a threshold value[23].

In the aggregate-based scheme, the following steps were involved in the neighborhood formation process:

1. Compute the cosine similarity between the current user navigation behavior vector (window *w*), and each of the row vectors of the association rules matrix
2. Select the most similar association rule row vector as the initial member of the neighborhood
3. Compute the cosine similarity between the current neighborhood, and each of the row vectors of the association rules matrix not currently selected as a member of the neighborhood
4. Select the most similar association rule row vector as the next member of the neighborhood
5. Aggregate the current neighborhood vector by adding to it URLS from the new member of the neighborhood determined in Step 4 above, that were not present in the neighborhood
6. Repeat Steps 3–5 until the cosine similarity falls below the threshold

[23] The similarity threshold was determined by selecting the value that resulted in minimum prediction error (also see Section 4.6, which deals with the evaluation of the collaborative filtering-based models).

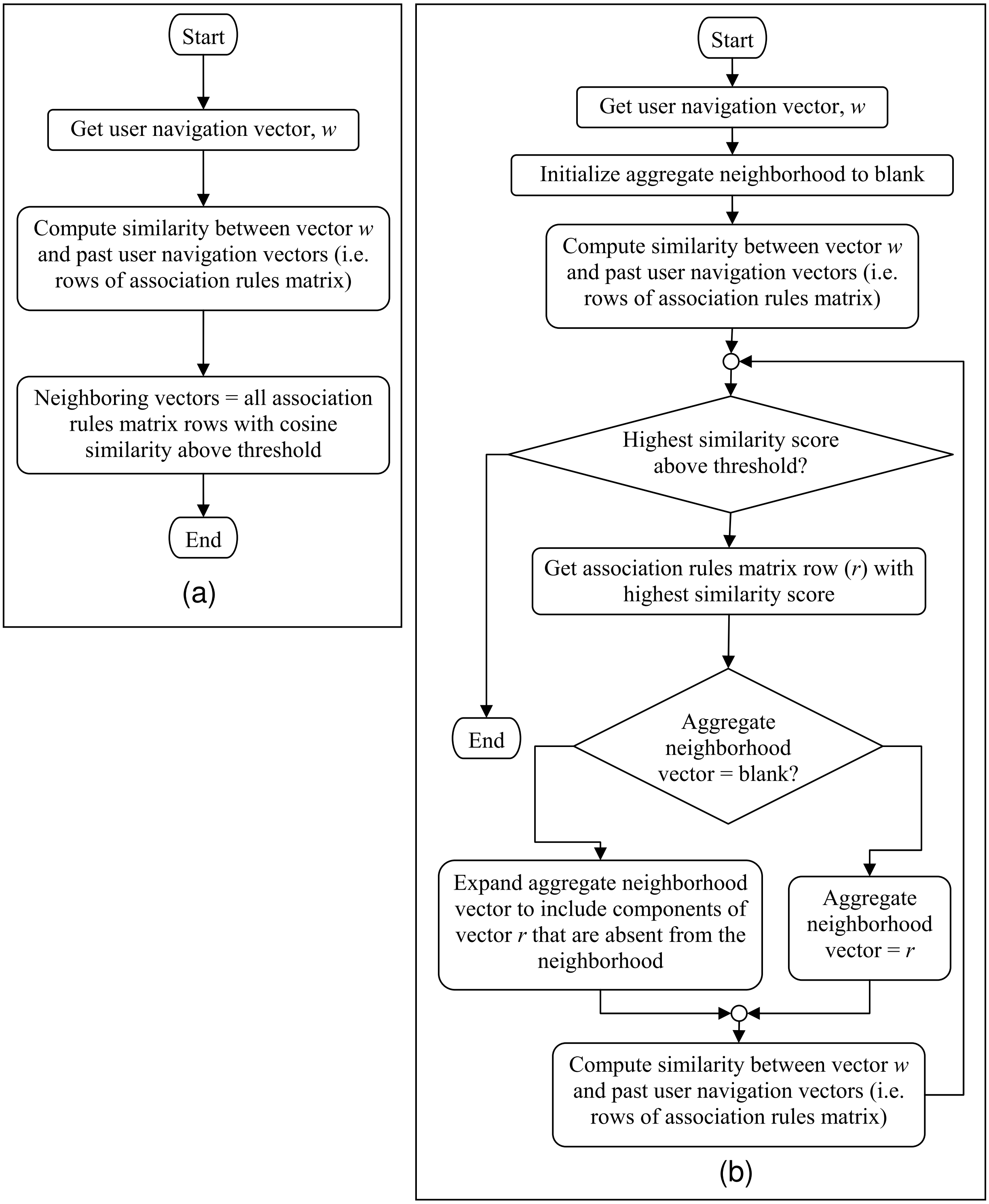

Start
Get user navigation vector, *w*
Compute similarity between vector *w* and past user navigation vectors (i.e. rows of association rules matrix)
Neighboring vectors = all association rules matrix rows with cosine similarity above threshold
End
(a)
Start
Get user navigation vector, *w*
Initialize aggregate neighborhood to blank
Compute similarity between vector *w* and past user navigation vectors (i.e. rows of association rules matrix)
Highest similarity score above threshold?
Get association rules matrix row (*r*) with highest similarity score
End
Aggregate neighborhood vector = blank?
Expand aggregate neighborhood vector to include components of vector *r* that are absent from the neighborhood
Aggregate neighborhood vector = *r*
Compute similarity between vector *w* and past user navigation vectors (i.e. rows of association rules matrix)
(b)

Figure 34. Summary of neighborhood formation processes for (a) center-based scheme, and (b) aggregate-based scheme.

The steps involved in determining the neighborhood vectors in both the center-based and aggregate-based schemes are summarized in Figure 34. As can be seen from the figure, the neighborhood formation process for the aggregate-based scheme is computationally very expensive. For that reason, during the evaluation of the prediction model using this scheme, the neighborhood size was not allowed to grow beyond 50.

Clearly in both the center-based scheme and the aggregate-based scheme, *k* was not a constant value as it depended on the number of past user navigation vectors whose similarity to the current user vector exceeded a threshold value.

Predicting Confidence Values (Ratings) for Recommendable Hyperlinks. For each neighborhood formation scheme, the confidence scores for the active user (*u*'s) rating for HTML Page *i*, was computed using 2 approaches: as the average of the *k* nearest neighbor ratings, and as a weighted average of the *k* nearest neighbor ratings for that page. The two formulas used were:

$$P_{u,i} = \frac{1}{N}\sum_{j=1}^{N} R_{j,i} \text{ for average similarity,} \qquad (6)$$

$$\text{and } P_{u,i} = \frac{1}{N}\sum_{j=1}^{N} R_{j,i} * Sim_{j,u}\text{, for weighted average similarity,} \qquad (7)$$

where, $P_{u,i}$ is User *u*'s predicted rating for page *i*, *N* is the neighborhood count, $R_{j,i}$ is the rating of page *i* for Vector *j* of the neighborhood, and $Sim_{j,u}$, the weighting factor, is the similarity between the navigation vectors of user *u* and Vector *j*.

4.6. Experiment 2: Evaluation of Performance of Collaborative Filtering Prediction Models

Experiment 2 was a simulation study designed to answer the following question: do the collaborative filtering models correctly predict the confidence values of an active user's interest in various web pages as the user navigates the website? If the models correctly predict these confidence values, then appropriate recommendations can be computed.

Data. The data used were the association rule matrix. The data were divided into a training and a test sets with 80% of the rows randomly assigned to the training set, and 20% to the test set.

Method. The general approach to evaluate performance was to use the training data alone to predict the test data, and use the predictions and prediction errors to determine the effectiveness of the prediction model. The evaluation involved three iterations, with each iteration involving a different set of training and test data, and the predictions of user interests averaged over the three iterations.

Center-based Neighborhood Formation Scheme. The navigation behavior vector for each row of the test data was used to simulate an active user's navigation behavior. The cosine similarity between each simulated user's navigation vector and each of the navigation vectors in the training set was then computed, and the *k* training vectors whose similarities to the current test vector were above a threshold selected as nearest neighbors. As explained in Section 4.5, *k* was a function of the similarity threshold chosen. The KNN vectors were computed for different similarity thresholds, and the threshold that resulted in optimal prediction errors selected.

Aggregate-based Neighborhood Formation Scheme. The approach used to evaluate the prediction model based on the aggregate-based prediction scheme was very similar to the method for the center-based scheme, except that we now took into consideration the much larger computation times required to generate the neighborhood. The following modifications were made:

- A random selection of only 250 data items from the test set was used, and predictions made using all of the training set data as before
- A maximum of 50 vectors were allowed in the neighborhood

5. RESULTS AND DISCUSSION

This section presents the data that were used in this study, as well as results of the implementation procedures described in Section 4. We begin by presenting statistics on the primary data used, namely web pages of the web site and the user access logs (Section 5.1). Section 5.2 follows with results of the web page classification study that were used to classify web pages as navigation or content pages. Section 5.3 summarizes the data that were used to construct past user profiles, against which current users could be compared. These data comprise past user sessions data, transactions data, and association rules derived from the transactions. Section 5.4 presents an evaluation of the newly proposed joinless apriori algorithm relative to the classical apriori algorithm; Section 5.5 an evaluation of the performance of the CLARANS clustering algorithm in high dimensional space and relative to the PAM and CLARA clustering algorithms on which CLARANS is based; and Section 5.6, evaluation of the effectiveness of the prediction models for current users' pages of interest as they navigate a website.

5.1. Web Pages and Access Logs

This section presents statistics on the number and contents of HTML pages in the website, as well as on the user access logs that were used in the study. Knowledge of the web pages was required to construct the web site topology, which as pointed out in Section 3.1, was required by one of the heuristic that were applied to deduce past user navigation paths. The PR, LW, and PR × ILW metrics of these pages were also useful in the determination of user navigation behaviors as explained in Section 3.2.1. The access logs on the other hand, contained past user navigation paths which were mined to determine the relationships between typical user navigation behaviors and the web pages they found interesting.

Web Page Statistics. 4491 HTML pages were discovered by the web crawler. Table 2 shows the following statistics of these pages: LW, PR, and PR × ILW frequencies.

Table 2. *Web site HTML page statistics: Links-to-Word count ratio (LW), Page Rank (PR), and Page Rank × Inverse Links-to-Word count ratio (PR × ILW).*

LW		PR		PR × ILW	
Range	Frequency	Range	Frequency	Range	Frequency
0<=LW<0.05	1575	0<=PR<0.15	733	0<=PR x ILW <0.15	24
0.05<=LW<0.10	919	0.15<=PR<0.20	1572	0.5<=PR x ILW <1.0	810
0.10<=LW<0.15	759	0.20<=PR<0.25	540	1.0<=PR x ILW <1.5	1057
0.15<=LW<0.20	351	0.25<=PR<0.30	207	1.5<=PR x ILW <2.0	489
0.20<=LW<0.25	162	0.30<=PR<0.35	147	2.0<=PR x ILW <2.5	312
0.25<=LW<0.30	144	0.35<=PR<0.40	119	2.5<=PR x ILW <3.0	204
0.30<=LW<0.35	168	0.40<=PR<0.45	80	3.0<=PR x ILW <3.5	168
0.35<=LW<0.40	198	0.45<=PR<0.50	66	3.5<=PR x ILW <4.0	181
0.40<=LW<0.45	117	0.50<=PR<0.55	70	4.0<=PR x ILW <4.5	178
0.45<=LW<0.50	31	0.55<=PR<0.60	66	4.5<=PR x ILW <5.0	124
0.50<=LW<0.55	15	0.60<=PR<0.65	62	5.0<=PR x ILW <5.5	117
0.55<=LW<0.60	8	0.65<=PR<0.70	62	5.5<=PR x ILW <6.0	85
0.60<=LW<0.65	2	0.70<=PR<0.75	40	6.0<=PR x ILW <6.5	75
0.65<=LW<0.70	0	0.75<=PR<0.80	35	6.5<=PR x ILW <7.0	55
0.70<=LW<0.75	3	0.80<=PR<0.85	33	7.0<=PR x ILW <7.5	68
0.75<=LW<0.80	12	0.85<=PR<0.90	24	7.5<=PR x ILW <8.0	56
0.80<=LW<0.85	6	0.90<=PR<0.95	30	8.0<=PR x ILW <8.5	41
0.85<=LW<0.90	0	0.95<=PR<1.0	18	8.5<=PR x ILW <9.0	38
0.90<=LW<0.95	2	1<=PR<2	252	9.0<=PR x ILW <9.5	35
0.95<=LW<1	19	2<=PR<3	169	9.5<=PR x ILW <10.0	43
		3<=PR<4	16	10.0<=PR x ILW <10.5	35
		4<=PR<5	13	10.5<=PR x ILW <11.0	16
		5<=PR<6	64	11.0<=PR x ILW <11.5	31
		6<=PR<7	5	11.5<=PR x ILW <12.0	23
		7<=PR<8	20	12.0<=PR x ILW <12.5	17
		8<=PR<9	6	12.5<=PR x ILW <13.0	12
		9<=PR<10	20	13.0<=PR x ILW <13.5	16
		10<=PR<15	7	13.5<=PR x ILW <14.0	15
		15<=PR<20	3	14.0<=PR x ILW <14.5	13
		20<=PR<25	6	14.5<=PR x ILW <15.0	10
		25<=PR<30	3	15.0<=PR x ILW <15.5	15
		30<=PR<35	1	15.5<=PR x ILW <16.0	3
		35<=PR<40	1	16.0<=PR x ILW <16.5	15
		40<=PR<45	0	16.5<=PR x ILW <17.0	3
		45<=PR<50	0	17.0<=PR x ILW <17.5	8
		50<=PR<55	1	17.5<=PR x ILW <18.0	5
				18.0<=PR x ILW <18.5	6
				18.5<=PR x ILW <19.0	5
				19.0<=PR x ILW <19.5	6
				19.5<=PR x ILW <20.0	3
				20<=PR x ILW <25	24
				25<=PR x ILW <30	20
				30<=PR x ILW <35	14
				35<=PR x ILW <40	4
				40<=PR x ILW <45	4
				45<=PR x ILW <50	4
				50<=PR x ILW <55	2
				55<=PR x ILW <60	1
				60<=PR x ILW <65	0
				65<=PR x ILW <70	0
				70<=PR x ILW <75	1

Web Page Terms. As explained in Section 4.1.1, the HTML pages of the website were parsed, the contents extracted, and the vector space model used to represent them. The dimensionality of the document vectors were reduced by stemming and elimination of low- and high-frequency terms. Table 3 shows the distribution of terms across the web pages before and after the elimination of high and low frequency terms.

There were a total of 25882 terms before high and low frequency terms were eliminated and 3027 terms after. The low frequency term threshold was 40 (i.e. all terms that occurred less than 40 times in the web site were eliminated – roughly equivalent to 1% of the web pages), while the high frequency threshold was 1000 (all terms that occurred more than 1000 times in the web site were eliminated – i.e., roughly equivalent to 25% of the web pages).

Table 3. *Distribution of terms in web pages (a) before and (b) after elimination of high and low frequency terms.*

(a)

term frequency	No of terms with		
range	frequency in range f		Cumulative
(f)	(t)	Percentage	percentage
1	7245	28.46	28.46
2	4629	11.74	40.20
3	1963	9.21	49.41
4	1636	7.30	56.72
5	926	4.62	61.34
5<f<=10	2782	11.26	72.59
10<f<=20	1991	8.66	81.26
20<f<=30	827	3.75	85.01
30<f<=40	516	2.21	87.22
40<f<=50	349	1.53	88.74
50<f<=60	237	1.11	89.85
60<f<=70	193	0.88	90.73
70<f<=80	169	0.73	91.47
80<f<=90	146	0.52	91.98
90<f<=100	92	0.49	92.48
100<f<=150	414	1.65	94.13
150<f<=200	242	1.10	95.23
200<f<=1000	813	3.49	98.72
1000<f<=10000	254	1.26	99.97

(b)

term frequency	No of terms with		
range	frequency in range f		Cumulative
(f)	(t)	Percentage	percentage
5<f<=50	445	14.70	14.70
5<f<=100	967	31.95	46.65
5<f<=150	427	14.11	60.75
5<f<=200	285	9.42	70.17
5<f<=250	170	5.62	75.78
5<f<=300	136	4.49	80.28
5<f<=350	99	3.27	83.55
5<f<=400	89	2.94	86.49
5<f<=450	63	2.08	88.57
5<f<=500	58	1.92	90.49
5<f<=550	44	1.45	91.94
5<f<=600	35	1.16	93.10
5<f<=650	43	1.42	94.52
5<f<=700	21	0.69	95.21
5<f<=750	39	1.29	96.50
5<f<=800	27	0.89	97.39
5<f<=850	17	0.56	97.95
5<f<=900	26	0.86	98.81
5<f<=950	19	0.63	99.44
5<f<=1000	17	0.56	100.00

User Access Log Data. These were the user access log for the website of SIS for the months of June to August 2004. The raw data were stored in the common log format (W3C, 2003) and totaled about 500MB.

5.2. Web Page Classification Study Results

As explained in Section 4.2.5, the objective of the web page classification study was to evaluate the utility of the LW, PR, and PR × ILW metrics in the classification of web pages as content or navigation. By determining appropriate thresholds for these metrics, it would be possible to break down very long transactions determined using the MFR heuristic into smaller, more desirable ones. Three users who were very familiar with the web site were asked to classify a sample of pages from the site as content or navigation. Figure 35 shows the page ratings on the content, navigation and dual content/navigation scales by each of the subjects (S1, S2, and S3). As can be seen from the figure, the ratings were largely consistent, except for some of the ratings by Subject S2, which we believed were errors (see comments in the last column of Figure 35). For these erroneous cases, the relevant ratings were not used in computing average ratings.

The variation of average user ratings with LW is shown in Figure 36. As expected, web pages were generally classified as content (high content and dual scale ratings) for low values of LW (left of graph), and as navigation (high navigation scale ratings) for high LW (right of graph). Navigation ratings were strongly inversely correlated with content ratings. We also notice from Figure 36 that ratings on the content scale closely mirror those on the dual scale. These results suggest that a single content scale is sufficient for this type of study.

The results presented in Figure 36 also suggest that LW alone is not sufficient to correctly classify content and navigation pages. For example, there are a number of pages with relatively high LW ratio that subjects classified as content pages, but that using the LW parameter alone, would be classified as navigation pages. We present next, results that examine the LW, PR, and PR × ILW metrics more closely.

Table 4 shows the mean content ratings, LW, PR, and PR × ILW values arranged first in order of LW, then in order of PR, and finally in order of PR × ILW. From the table, it can be seen that in general, web pages with a LW value below 0.05 or a PR × ILW value above 1.8 can be classified as content pages. PR alone, however, does not afford such a dividing line that separates content from navigation pages.

	Content Scale			Navigation Scale			Dual Content & Navigation Scale			Comments	
Item No	S1	S2	S3	S1	S2	S3	S1	S2	S3		
1	5	5	5	5	5	3	5	10	6	**d**	S2: dual rating inconsistent
2	8	3	4	2	7	4	7	2	5		
3	10	10	10	0	0	0	10	10	10		
4	9	7	7	1	2	7	9	6	5		
6	2	10	2	8	0	8	2	10	2	**a**	S2: Page comprises moderate amount of text, followed by a long list of hyperlinks at bottom which S2 must have missed
7	9	8	7	2	1	2	8	8	7		
8	8	8	4	3	1	6	7	7	4		
9	1	8	1	6	1	10	3	8	1	**a**	S2: Page comprises almost entirely of hyperlinks
10	9	2	9	7	2	8	8	2	9	**c**	S2: Content rating too low; page comprises mainly plain text and a few hyperlinks at the top and bottom of the page
11	1	1	1	10	9	10	0	1	0		
12	10	7	9	1	1	1	9	7	10		
13	10	8	9	3	1	3	8	8	9		
14	8	8	7	7	1	4	5	8	7		
15	10	10	10	0	0	0	10	10	10		
16	9	10	10	2	0	1	9	10	9		
17	9	10	10	1	0	1	9	10	10		
18	5	10	6	8	0	4	4	10	5	**a**	S2: All ratings by S2 inconsistent with the page characteristics; page is a large table with several hyperlinks and a moderate amount of text
19	9	9	6	2	1	3	9	9	7		
20	9	9	8	2	1	2	9	9	8		
21	8	10	7	3	0	7	8	10	5		
22	7	10	9	2	0	2	8	10	9		
23	10	9	5	0	0	1	9	9	9		
24	10	9	9	0	0	1	10	9	10		
25	1	10	1	9	0	9	1	10	0	**a**	S2: Ratings inconsistent with the page characteristic; page has a high number of hyperlinks relative to page contents
26	1	10	1	9	0	10	1	10	0	**a**	S2: Ratings inconsistent with the page characteristic; page has a high number of hyperlinks relative to page contents
27	9	10	8	1	0	2	9	10	9		
28	10	5	7	2	5	7	9	5	5		
30	9	5	2	2	5	8	8	5	1		
31	10	10	10	0	0	0	10	10	10		
32	7	6	7	3	4	3	6	6	7		
33	5	10	2	9	0	3	2	10	3		
34	9	8	10	1	1	2	9	8	10		
35	9	9	10	1	1	3	9	9	9		

a – all 3 of subject's ratings not used in computing average score
d – subject's dual scale rating not used in computing average score
c - subject's content scale rating not used in computing average score

Figure 35. Subject ratings of the web pages presented in User Study 1. The commented items refer to cases where there were large discrepancies in subject ratings. Results of 3 out of the 36 web pages that were presented to user are omitted because these pages were no longer linked to by the time the results were analyzed.

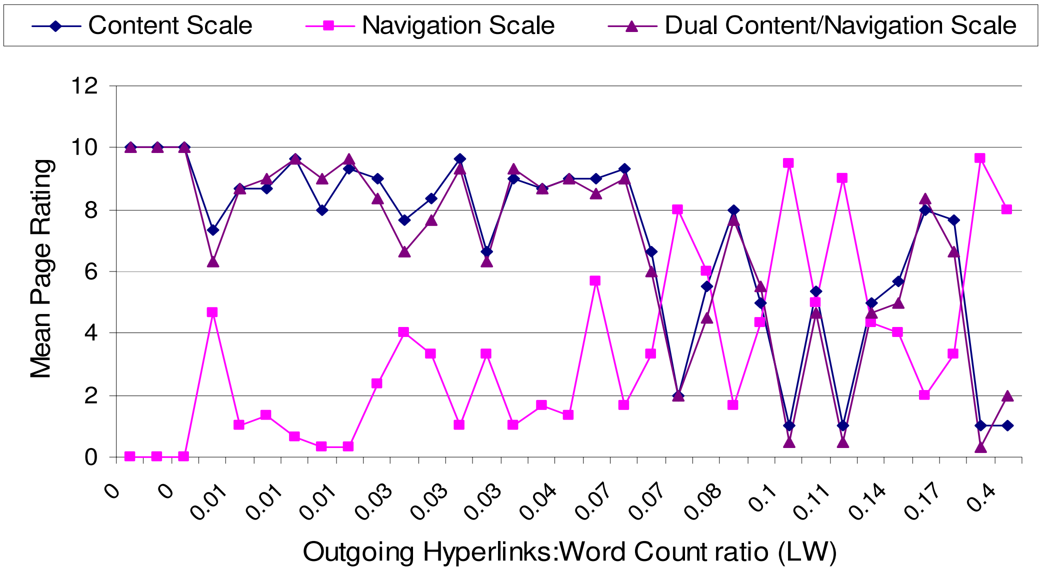


Figure 36. Variation of page ratings with LW on the content, navigation and dual content/navigation ratings scales.

Table 4. *Combining LW and PR × ILW thresholds to classify web pages as content or navigation.*

Sorted by LW					Sorted by PR					Sorted by PR×ILW				
	Mean					Mean					Mean			
Page	Content				Page	Content				Page	Content			
ID	Rating	LW	PR	PR×ILW	ID	Rating	LW	PR	PR×ILW	ID	Rating	LW	PR	PR×ILW
31	10.00	0.00	0.16	6.20	2	5.00	0.14	7.65	7.30	2	5.00	0.14	7.65	7.30
15	10.00	0.00	0.19	6.14	8	6.67	0.07	2.04	5.55	31	10.00	0.00	0.16	6.20
3	10.00	0.00	0.15	2.63	4	7.67	0.17	1.85	3.27	15	10.00	0.00	0.19	6.14
28	7.33	0.01	0.18	3.80	1	5.00	0.09	1.12	3.48	12	8.67	0.01	0.41	5.95
12	8.67	0.01	0.41	5.95	35	9.33	0.07	0.80	3.44	23	8.00	0.01	0.18	5.84
22	8.67	0.01	0.18	3.77	11	1.00	0.34	0.78	1.51	8	6.67	0.07	2.04	5.55
17	9.67	0.01	0.17	3.86	7	8.00	0.08	0.77	3.12	24	9.33	0.01	0.17	5.05
23	8.00	0.01	0.18	5.84	34	9.00	0.04	0.63	3.88	13	9.00	0.02	0.34	4.68
24	9.33	0.01	0.17	5.05	9	1.00	0.40	0.60	1.23	34	9.00	0.04	0.63	3.88
13	9.00	0.02	0.34	4.68	10	9.00	0.05	0.56	3.37	17	9.67	0.01	0.17	3.86
14	7.67	0.03	0.23	2.65	6	2.00	0.07	0.53	2.80	28	7.33	0.01	0.18	3.80
21	8.33	0.03	0.17	2.57	12	8.67	0.01	0.41	5.95	22	8.67	0.01	0.18	3.77
16	9.67	0.03	0.18	2.61	13	9.00	0.02	0.34	4.68	1	5.00	0.09	1.12	3.48
32	6.67	0.03	0.15	2.15	30	5.33	0.11	0.29	1.62	35	9.33	0.07	0.80	3.44
27	9.00	0.03	0.20	2.58	14	7.67	0.03	0.23	2.65	10	9.00	0.05	0.56	3.37
20	8.67	0.04	0.15	1.85	27	9.00	0.03	0.20	2.58	4	7.67	0.17	1.85	3.27
34	9.00	0.04	0.63	3.88	15	10.00	0.00	0.19	6.14	7	8.00	0.08	0.77	3.12
10	9.00	0.05	0.56	3.37	23	8.00	0.01	0.18	5.84	6	2.00	0.07	0.53	2.80
35	9.33	0.07	0.80	3.44	28	7.33	0.01	0.18	3.80	14	7.67	0.03	0.23	2.65
8	6.67	0.07	2.04	5.55	22	8.67	0.01	0.18	3.77	3	10.00	0.00	0.15	2.63
6	2.00	0.07	0.53	2.80	16	9.67	0.03	0.18	2.61	16	9.67	0.03	0.18	2.61
*18	5.50	0.08	0.15	1.36	24	9.33	0.01	0.17	5.05	27	9.00	0.03	0.20	2.58
7	8.00	0.08	0.77	3.12	17	9.67	0.01	0.17	3.86	21	8.33	0.03	0.17	2.57
1	5.00	0.09	1.12	3.48	21	8.33	0.03	0.17	2.57	32	6.67	0.03	0.15	2.15
26	1.00	0.10	0.16	1.28	25	1.00	0.11	0.17	1.21	20	8.67	0.04	0.15	1.85
*30	5.33	0.11	0.29	1.62	33	5.67	0.14	0.17	1.08	*30	5.33	0.11	0.29	1.62
25	1.00	0.11	0.17	1.21	31	10.00	0.00	0.16	6.20	11	1.00	0.34	0.78	1.51
2	5.00	0.14	7.65	7.30	26	1.00	0.10	0.16	1.28	*18	5.50	0.08	0.15	1.36
*33	5.67	0.14	0.17	1.08	3	10.00	0.00	0.15	2.63	26	1.00	0.10	0.16	1.28
*19	8.00	0.16	0.15	0.97	32	6.67	0.03	0.15	2.15	9	1.00	0.40	0.60	1.23
4	7.67	0.17	1.85	3.27	20	8.67	0.04	0.15	1.85	25	1.00	0.11	0.17	1.21
11	1.00	0.34	0.78	1.51	18	5.50	0.08	0.15	1.36	*33	5.67	0.14	0.17	1.08
9	1.00	0.40	0.60	1.23	19	8.00	0.16	0.15	0.97	*19	8.00	0.16	0.15	0.97

Using the LW and PR × ILW thresholds mentioned above, we notice from Table 4 that using LW alone, content pages 35, 8, 7, 1, 2, and 4 (shaded) would normally be classified as navigation pages because their LW values are above the minimum threshold of 0.05. However, because the corresponding PR × ILW values are above the 1.8 threshold value for classifying pages as content pages, these pages can be correctly classified using the PR × ILW metric. On the other hand, navigation Page 6 which was correctly classified using LW alone is now incorrectly classified using PR × ILW as a content page.

The shaded pages marked with an asterisk, "*" (30, 18, 33, and 19) could not be correctly classified as content pages. We consider each of them in turn. Pages 18, 30 and 33 happen to be borderline content pages, and they happen to have only 2 and 1 incoming links respectively. Page 19 on the other hand was rated very highly as a content page by subjects, but both LW and PR × ILW failed to classify it as a content page. This page too is linked to from only one other page in the website. In all 4 cases mentioned above, the PR score was too low (because they had too few incoming links) to have made a large impact on the PR × ILW score and so the pages continued to be classified as navigation. Because these pages are linked to from so few pages, web users have very few occasions to ever reach them, and so not treating them as content or potentially recommendable pages is actually desirable.

5.3. User Profiling Data

This section presents a summary of the distribution of user sessions that were extracted from the user access logs, the transactions (or user behavior vectors) that were extracted from the sessions, and the distribution of association rules mined from the transactions data, for various levels of the minimum rule support count.

Sessions and Transactions. Statistics on the number and size distribution of unique user sessions that were extracted from the user access log are presented in Table 5, while Table 6 shows the distribution of transactions following applications of the MFR heuristic, and the PR × ILW heuristic with LW threshold of 0.05 and PR × ILW threshold of 1.8. As can be seen from Table 6, the PR × ILW heuristic resulted in a reduction of the average transaction vector length, a desirable property for the application of association rule mining algorithms.

Association Rules. Both the joinless and classical apriori algorithms yielded the same association rules as expected, and so there was no question of evaluating accuracy. Table 7 shows the number of association rules discovered, and the number of rows of the association rules matrix for LW and PR × ILW thresholds of 0.05 and 1.8 respectively, and for various values of minimum support count.

Table 5. *Distribution of sessions discovered in user access logs*.

Session length	Count	Session length	Count	Session length	Count
1	269486	13	277	41-45	68
2	13496	14	224	46-50	53
3	7516	15	185	51-75	106
4	4348	16	163	76-100	49
5	2775	17	144	101-200	51
6	1978	18	101	201-300	31
7	1372	19	105	301-400	47
8	974	20	64	401-500	20
9	701	21-25	289	501-750	33
10	576	26-30	177	751-1000	59
11	421	31-35	119	1001-2000	75
12	349	36-40	83	2001-3000	3
Session Count: 306513					
Mean session Length: 1.31					

Table 6. *Distribution of transactions discovered in user sessions using MFR alone and both MFR and PR × ILW for PR and PR × ILW thresholds of 0.05 and 1.8*.

Transaction Length	Using MFR heuristic	Using MFR and PR × ILW
2	12041	180377
3	7461	34203
4	4296	11183
5	2805	4926
6	1806	2735
7	1341	492
8	868	226
9	615	155
10	536	118
11–15	1363	234
16–20	678	105
21–25	510	78
26–30	587	51
31–40	804	58
41–50	551	33
51–75	909	4
76–100	359	0
101–150	222	0
151–200	114	0
Total number of transactions	37866	234979
Mean transaction length	9.53	2.42

Table 7. *Distribution of association rule count and association rule matrix size for various values of minimum support count*

	Minimum support count						
	1	2	12	24	59	118	235
No of association rules	99426	86477	50120	41046	33403	26285	16293
No of rows in association rules matrix	50094	37145	7374	3691	1678	927	388

5.4. Evaluation of Joinless Apriori Algorithm

One of the contributions of this research has been the development of a new apriori algorithm that avoids the join step in the classical apriori algorithm. In Section 4.3, it was demonstrated that both the classical and the new algorithms are equivalent in that they result in the same association rules, and a procedure to evaluate the new algorithm by comparing it with the classical algorithm was presented in Section 4.3.4. This section presents the evaluation results.

The distributions of execution time and minimum support count for the joinless and classical apriori algorithms are shown in Figure 37 ((a) and (b)). The corresponding variation in average rule length with minimum support count is shown in Figure 37c. Figure 38 shows the typical variation of average execution time per rule with rule length. Figure 39 shows the typical variation of total execution time as a function of rule length for the joinless and classical apriori algorithms.

As can be seen from Figure 37 (a), and (b), for both the joinless and classical apriori algorithms, the execution time increases with average association rule length (see each of the curves in the figure); this is expected. In the joinless algorithm however, the execution time is fairly constant for different values of minimum rule support count and the same average rule length. Intuitively, one would expect the execution time to decrease with increase in the minimum support count (as is the case with the classical algorithm (Figure 37 (b)), since larger support threshold translate to fewer rules generated. We also notice from Figure 37 (a) and (b) that the joinless apriori algorithm performs much better than the classical algorithm for small values of minimum support count when a large number of rules are generated (see execution times), but the classical algorithm performs slightly better for large values of minimum support count (when a smaller number of rules are generated). The reason for this is related to the fact that the join step is a critical factor in the classical algorithm: the smaller the number of rules generated, the smaller the sizes of tables resulting from the join, and the better the performance.

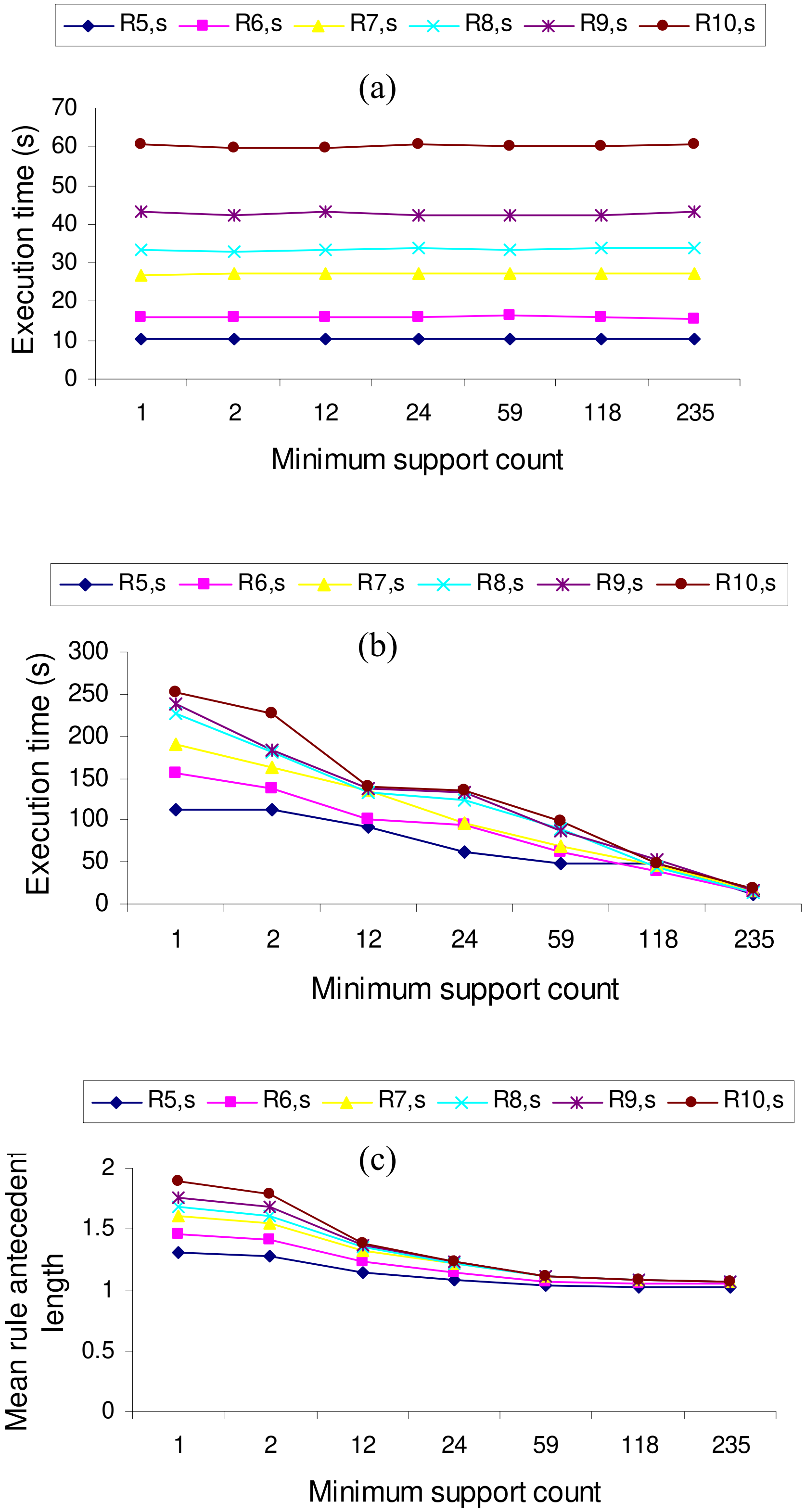


Figure 37. Variation of execution time with minimum support count for (a) joinless apriori algorithm and (b) classical apriori algorithm. Vaiation of corresponding average rule length with minimum support count (c).

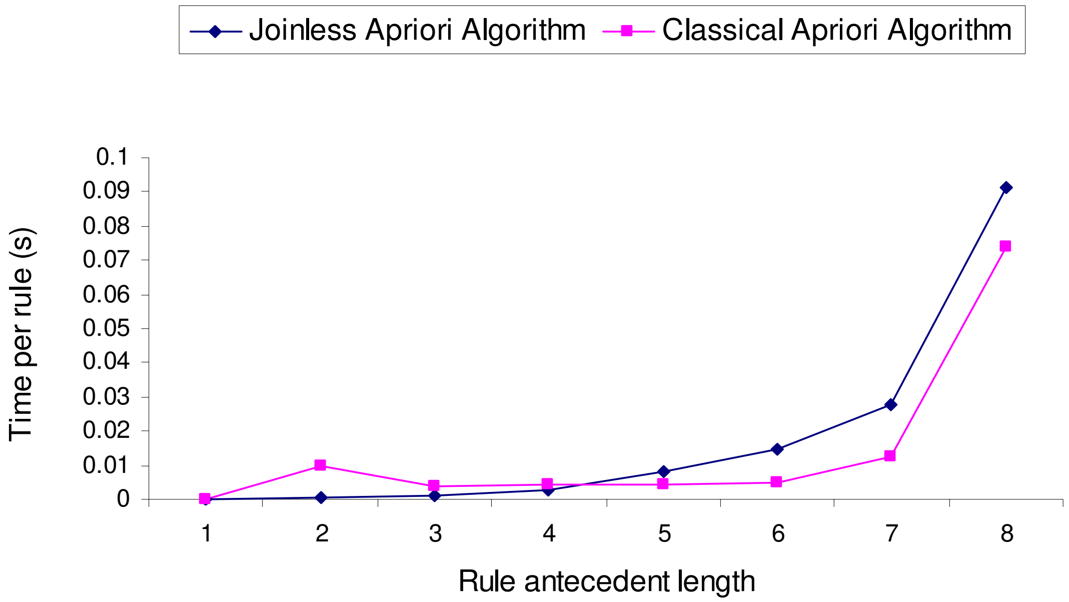


Figure 38. Variation of average generation time per rule with rule length for $R_{10,1.}$

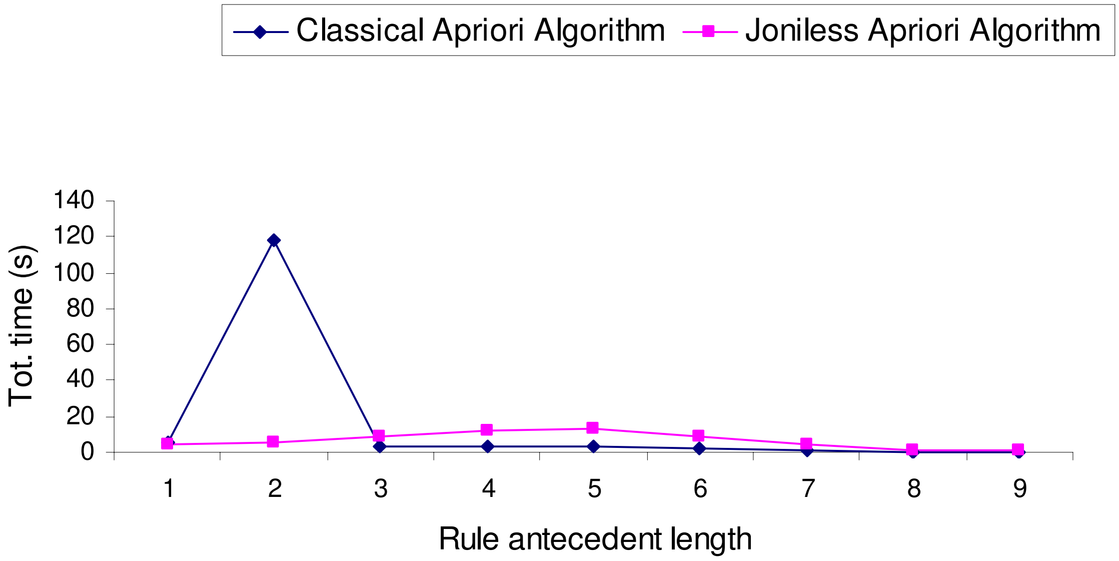


Figure 39. Variation of execution time with rule length for $R_{10,1.}$

For both the joinless and classical apriori algorithms, the time it takes to generate a single association rule increases exponentially with the length of the rule (see Figure 38). This may initially suggest that both algorithms face an explosion in computational time as longer rules are generated. This is however, not the case as we saw in Figure 37 (a) and (b). The reason for this is the fact that number of these expensive, longer rules is small.

The relative time efficiencies of the joinless and classical apriori algorithms for different rule lengths are illustrated in Figure 39. As can be seen from the figure, the classical apriori algorithm performs much worse than the joinless algorithm when the rule length is 2, while the performance of the classical apriori algorithms is slightly better for higher rule lengths. The shape of the classical apriori algorithm curve can be explained as follows: the first pass of the algorithm is relatively inexpensive, involving simply a count of 1-itemsets in the database; the second pass is very expensive because it involves a join; subsequent passes turn out to be progressively less expensive because the joins progressively become relatively inexpensive as the apriori property is used to prune the input to the join.

5.5. Evaluation of CLARANS in High Dimensional Space

In this section, we present an evaluation of the CLARANS clustering algorithm. We recall that the data that were used were simulated documents in high-dimensional document spaces. The procedure to generate these documents, which were divided into 2 groups R′ and R″ (respectively with high and low intra-cluster similarities) is outlined in Section 4.4.4. The evaluation is done in two stages. In the first stage, we calibrate the algorithm for the available data by determining optimal values for the two CLARANS parameters: *maxneighbors*, the number of neighboring nodes that need to be checked for a node with a lower cost before the algorithm concludes that the current node is a local minimum, and *numlocal*, the number of times the algorithm should iterate before concluding that the current set of medoids is the global minimum (Section 5.5.1).

Following the calibration of CLARANS, we present evaluation results on the performance of the algorithm (Section 5.5.2). We nwxt illustrate the behavior of CLARANS on the datasets under study, and then compare the efficiency and effectiveness of CLARANS with respect to the PAM and CLARA clustering algorithms.

5.5.1. Calibration of CLARANS

Parameter maxneighbors. The distribution of mean execution time and mean cluster dissimilarity for different values of parameter *maxneighbors* in the CLARANS algorithm, using data sets $R'_{1100,k}$ and $R'_{4500,k}$, is shown in Figure 40. Figure 41 shows the same results for data sets data sets $R''_{1100,k}$ and $R''_{4500,k}$. Parameter *numlocal* was set to a high value of 10, and so the execution times and cluster dissimilarities were averaged over 10 iterations of the algorithm. The number of neighboring nodes examined was not allowed to exceed 10000 because of the computational costs that would be involved otherwise; that accounts for the leveling off of execution times for data sets $R'_{4500,k}$ and $R''_{4500,k}$.

As can be seen from these results, the trend in cluster dissimilarities experiences only a slight decrease with increasing values of parameter *maxneighbors*, while execution time increases exponentially. So the main factor in the choice of parameter *maxneighbors* was execution time. The value 3.5% of total number of neighboring nodes to any medoid (*k*(*n*-*k*), which evaluates to a maximum value of 3.5% of 60(4500-60), a value close to 10000), was chosen as optimal.

Parameter numlocal. The distribution of mean execution time and mean cluster dissimilarity for various values of parameter *numlocal* for a fixed value of parameter *maxneighbors* set to 3.5% *k*(*n*-*k*), using data sets $R'_{1100,k}$ and $R'_{4500,k}$, is shown in Figure 42. Figure 43 shows the same results for data sets data sets $R''_{1100,k}$ and $R''_{4500,k}$. As in the case of determination of parameter *maxneighbors*, the main factor in the choice of parameter *maxneighbors* was execution time. A value of 2 for parameter *numlocal* was judged to be appropriate first because it is greater than 1 (meaning the algorithm has the opportunity for at least a second iteration to search for better clusters, and second because the cluster dissimilarity values do not change much for larger values of parameter *numlocal*.

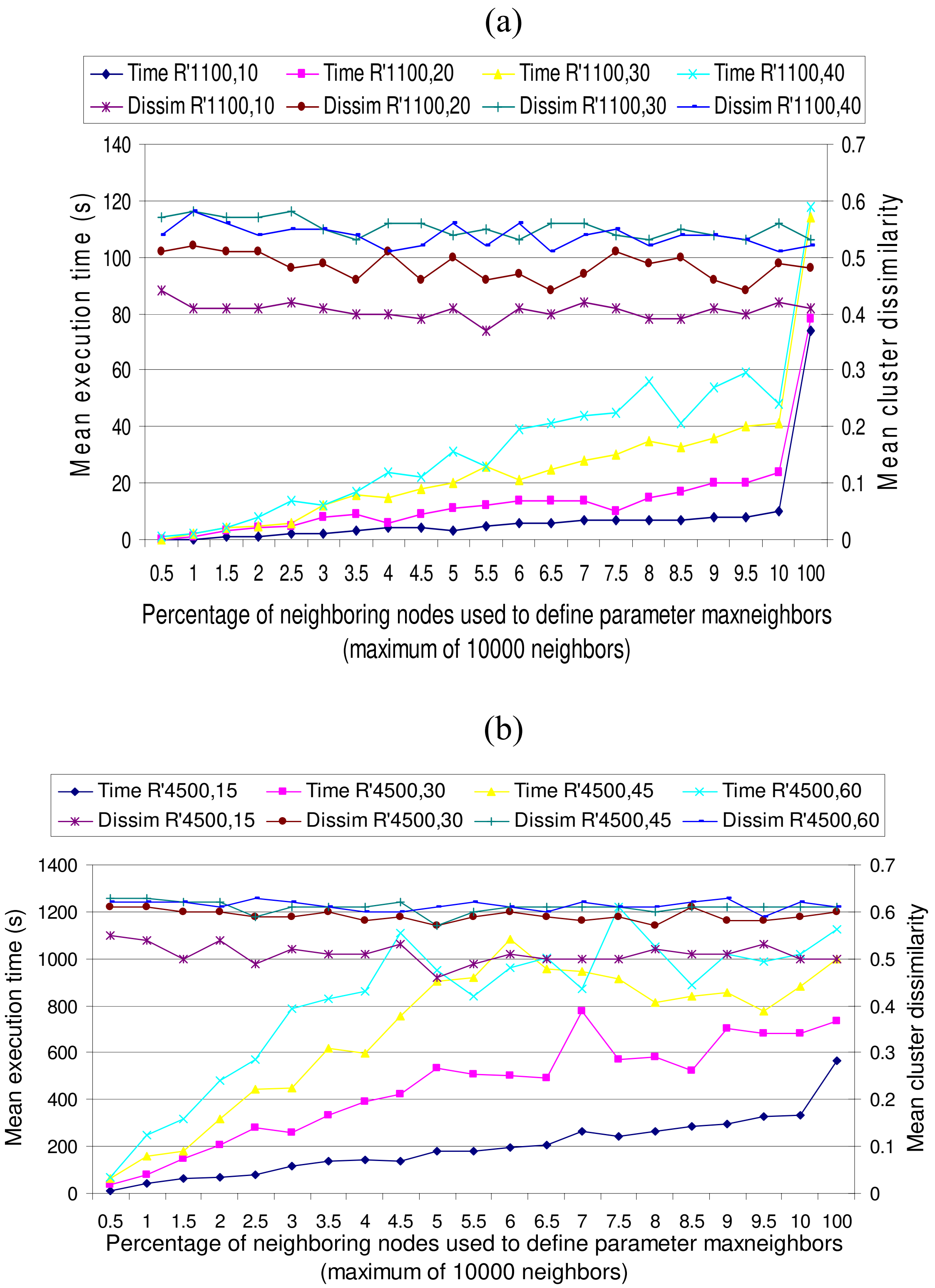


Figure 40. Determination of CLARANS parameter *maxneighbors*. Variation of execution time and mean cluster dissimilarity with percentage of neighboring nodes checked (parameter numlocal set to 10) for: (a) dataset $R'_{1100,k}$; (b) dataset $R'_{4500,k}$. The execution times in the second graph level off when the maximum value of *maxneighbors* (10000) is encountered.

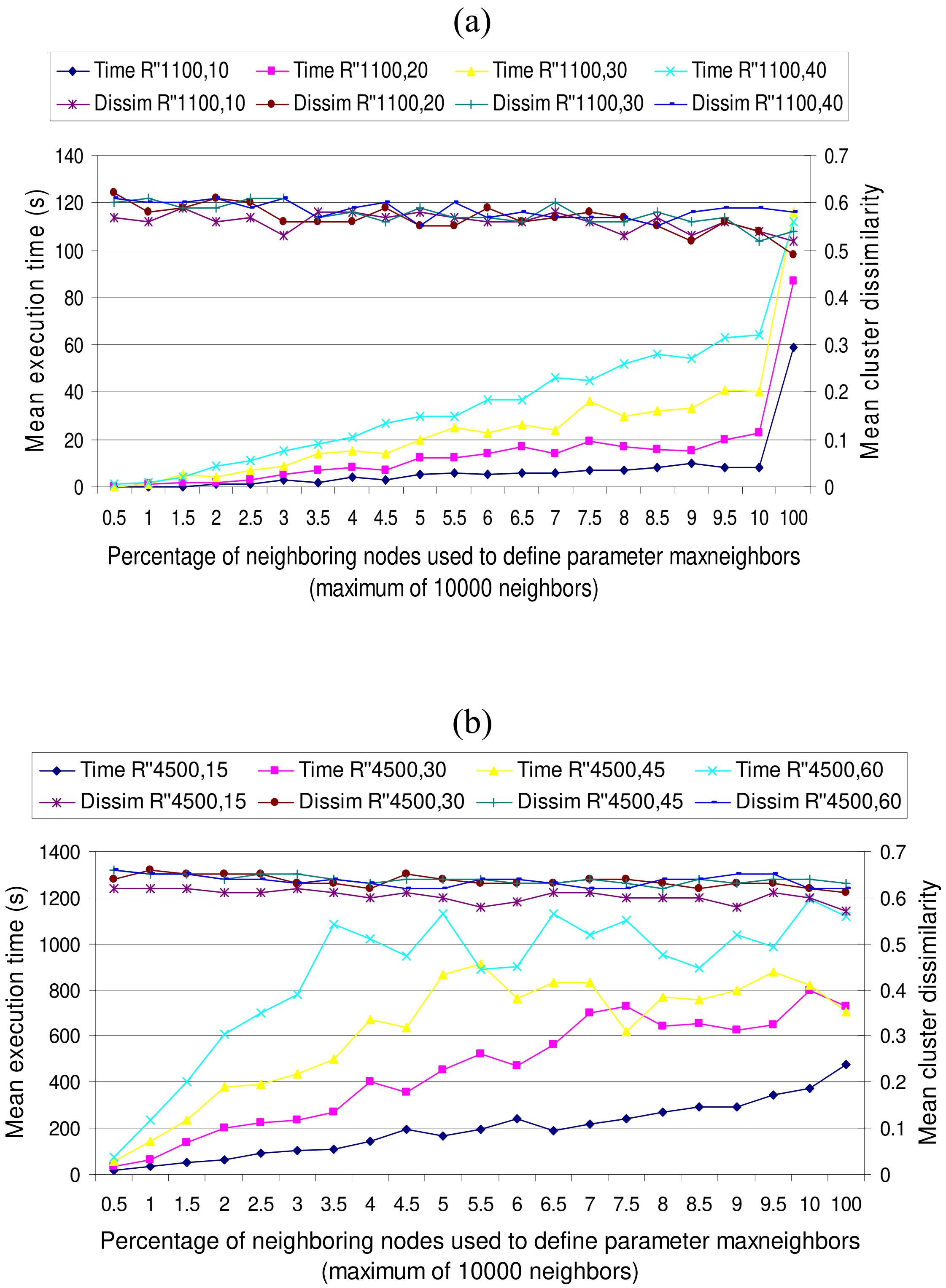
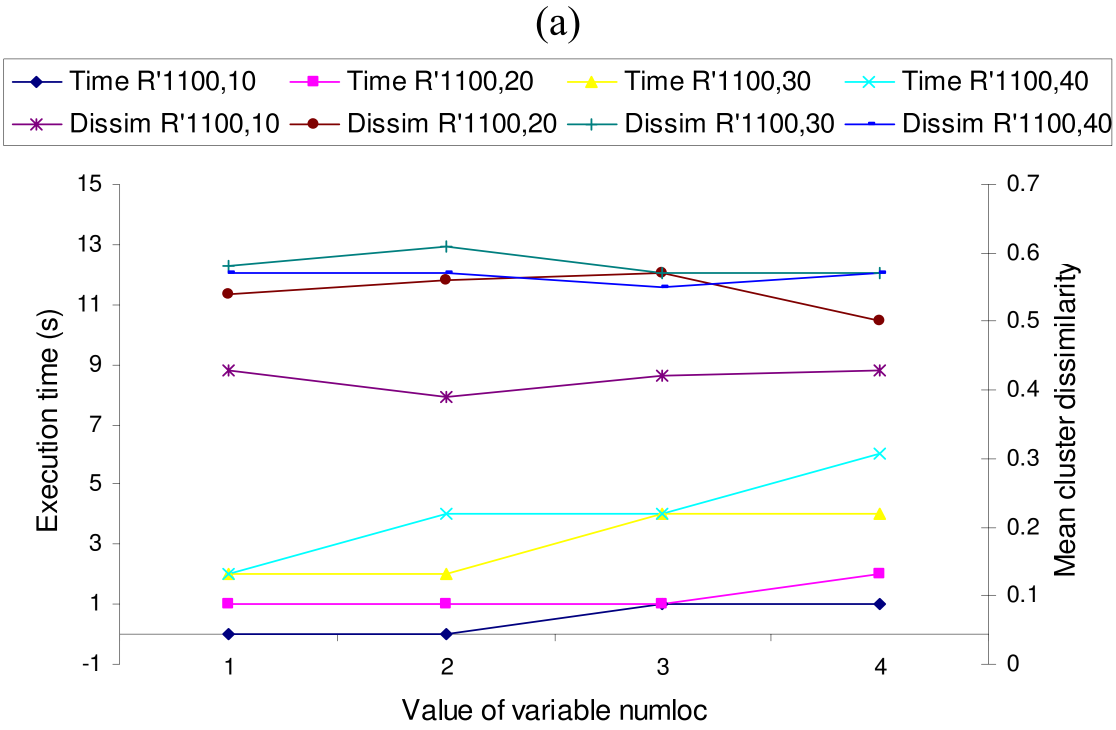


Figure 41. Determination of CLARANS parameter *maxneighbors*. Variation of execution time and mean cluster dissimilarity with percentage of neighboring nodes checked (parameter numlocal set to 10) for: (a) dataset $R''_{1100,k}$; (b) dataset $R''_{4500,k}$. The execution times in the second graph level off when the maximum value of maxneighbors (10000) is encountered.



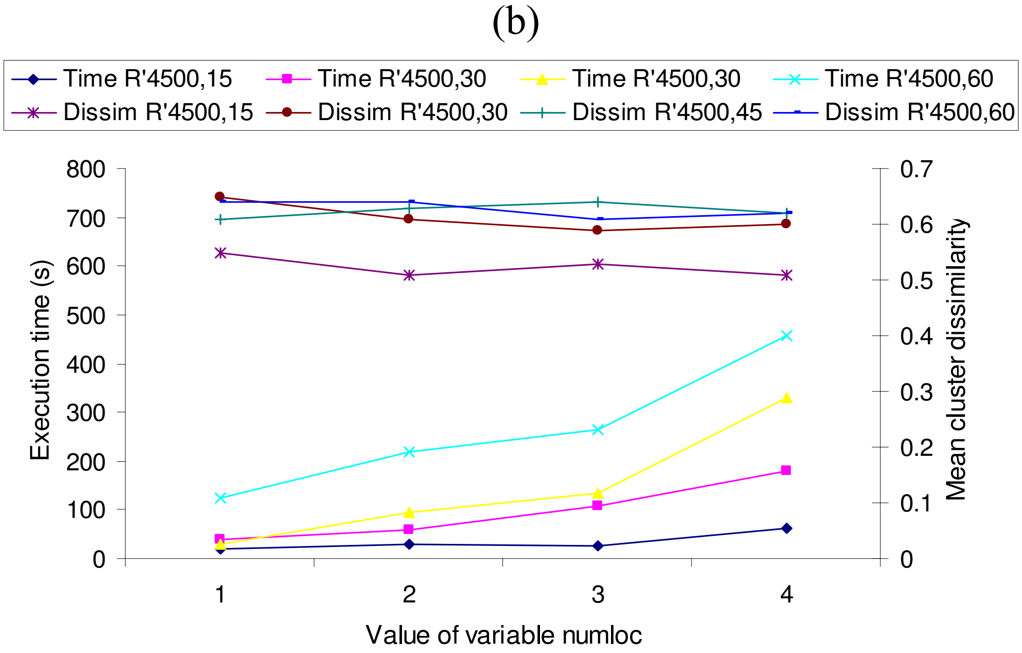


Figure 42. Variation of execution time and mean cluster dissimilarity, versus *numlocal* (parameter *maxneighbors* set to 3.5% of number of neighboring nodes – *k(n-k)*) for: (a) dataset R′$_{1100,k}$; (b) dataset R′$_{4500,k}$.

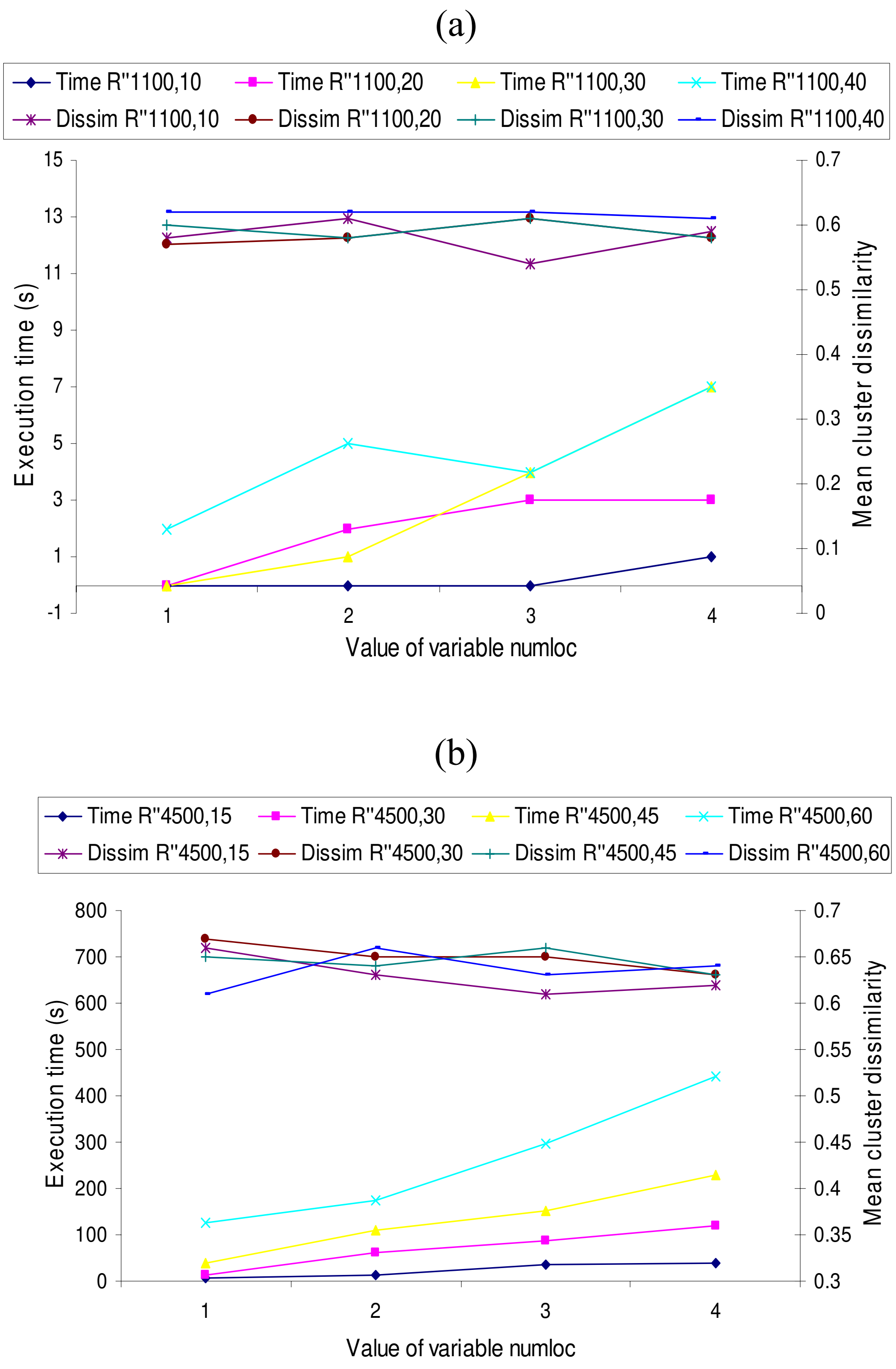


Figure 43. Variation of execution time and mean cluster dissimilarity, versus *numlocal* (parameter *maxneighbors* set to 3.5% of number of neighboring nodes – *k(n-k)*) for: (a) dataset R″$_{1100,k}$; (b) dataset R″$_{4500,k}$.

5.5.2. Performance of CLARANS

Performance on Data Sets. A comparison of the efficiency (execution times) and effectiveness (average dissimilarity values) of CLARANS on the datasets $R'_{1100,k}$, $R'_{4500,k}$, $R''_{1100,k}$, and $R''_{4500,k}$ is presented in Table 8. The following observations can be made from the results.

- The quality of clusters discovered is comparable for different values of the *maxneighbors* parameter
- CLARANS is more effective (lower dissimilarity scores) when the clusters are more distinct (datasets $R'_{n,k}$) than when they are not (datasets R″n,k), and more so for smaller values of *n* and *k*
- The times required to generate clusters increases for increasing values of parameter *maxneighbors* (this is expected) and are comparable for datasets $R'_{n,k}$ and datasets $R''_{n,k}$

Comparison of CLARANS to PAM and CLARA

Effectiveness. The distributions of average dissimilarities of discovered clusters using CLARA, CLARANS and PAM on datasets $R'_{1100,k}$ and $R''_{1100,k}$ are presented in Figure 44 (a) and using data sets $R'_{4500,k}$ and $R''_{4500,k}$, in Figure 44 (b). The distributions suggest that CLARANS is more effective (lower dissimilarity scores) than both CLARA and PAM; the effectiveness of CLARA and PAM on the other hand are about equal.

Efficiency. Table 9 shows the execution times of CLARA, CLARANS and PAM on datasets $R'_{n,k}$ and $R''_{n,k}$. For each of the algorithms, execution time increases with *n* and with *k*. Comparing efficiency across algorithms, we see that CLARA is much better than both CLARANS and PAM; CLARANS in turn is much better than PAM.

Table 8. *Comparison of performance (execution times and average cluster dissimilarities) of CLARANS on datasets $R'_{n,k}$ and $R''_{n,k}$.*

Maxneighbors (%)	Number of clusters (k)								Number of clusters (k)							
	k=10		k=20		k=30		k=40		k=15		k=30		k=45		k=60	
	Time (s)	Avg. Dissim.	Time (s)	Avg. Dissim.	Time (s)	Avg. Dissim.	Time (s)	Avg. Dissim.	Time (s)	Avg. Dissim.	Time (s)	Avg. Dissim.	Time (s)	Avg. Dissim.	Time (s)	Avg. Dissim.
	$R'_{1100,k}$								$R'_{4500,k}$							
0.5	0	0.44	0	0.51	0	0.57	1	0.54	13	0.55	35	0.61	66	0.63	70	0.62
1	0	0.41	1	0.52	2	0.58	2	0.58	44	0.54	80	0.61	161	0.63	249	0.62
1.5	1	0.41	3	0.51	4	0.57	4	0.56	61	0.5	148	0.6	181	0.62	318	0.62
2	1	0.41	4	0.51	5	0.57	8	0.54	70	0.54	207	0.6	319	0.62	482	0.61
2.5	2	0.42	5	0.48	6	0.58	14	0.55	80	0.49	279	0.59	445	0.59	569	0.63
3	2	0.41	8	0.49	12	0.55	12	0.55	117	0.52	260	0.59	447	0.61	786	0.62
3.5	3	0.4	9	0.46	16	0.53	17	0.54	135	0.51	331	0.6	618	0.61	828	0.61
4	4	0.4	6	0.51	15	0.56	24	0.51	142	0.51	390	0.58	598	0.61	863	0.6
4.5	4	0.39	9	0.46	18	0.56	22	0.52	139	0.53	422	0.59	754	0.62	1109	0.6
5	3	0.41	11	0.5	20	0.54	31	0.56	178	0.46	534	0.57	901	0.57	953	0.61
5.5	5	0.37	12	0.46	26	0.55	26	0.52	177	0.49	506	0.59	921	0.6	840	0.62
6	6	0.41	14	0.47	21	0.53	39	0.56	198	0.51	502	0.6	1081	0.61	962	0.61
6.5	6	0.4	14	0.44	25	0.56	41	0.51	208	0.5	492	0.59	954	0.61	1006	0.6
7	7	0.42	14	0.47	28	0.56	44	0.54	264	0.5	777	0.58	945	0.61	872	0.62
7.5	7	0.41	10	0.51	30	0.54	45	0.55	242	0.5	572	0.59	913	0.61	1221	0.61
8	7	0.39	15	0.49	35	0.53	56	0.52	263	0.52	583	0.57	814	0.6	1049	0.61
8.5	7	0.39	17	0.5	33	0.55	41	0.54	283	0.51	522	0.61	838	0.61	886	0.62
9	8	0.41	20	0.46	36	0.54	54	0.54	294	0.51	705	0.58	857	0.61	1021	0.63
9.5	8	0.4	20	0.44	40	0.53	59	0.53	329	0.53	680	0.58	779	0.61	988	0.59
10	10	0.42	24	0.49	41	0.56	48	0.51	335	0.5	684	0.59	883	0.61	1020	0.62
100	74	0.41	78	0.48	114	0.53	118	0.52	567	0.5	732	0.6	998	0.61	1124	0.61
	$R''_{1100,k}$								$R''_{4500,k}$							
0.5	0	0.57	0	0.62	0	0.6	1	0.61	19	0.62	37	0.64	60	0.66	76	0.66
1	0	0.56	1	0.58	1	0.61	2	0.6	33	0.62	61	0.66	146	0.65	235	0.65
1.5	0	0.59	2	0.59	5	0.59	4	0.6	54	0.62	136	0.65	236	0.65	400	0.65
2	1	0.56	2	0.61	4	0.59	9	0.61	65	0.61	199	0.65	376	0.64	608	0.64
2.5	1	0.57	3	0.6	7	0.61	11	0.59	92	0.61	225	0.65	392	0.65	699	0.64
3	3	0.53	5	0.56	9	0.61	15	0.61	106	0.62	237	0.63	434	0.65	779	0.63
3.5	2	0.58	7	0.56	14	0.57	18	0.57	107	0.61	268	0.63	499	0.64	1083	0.64
4	4	0.58	8	0.56	15	0.58	21	0.59	141	0.6	404	0.62	669	0.63	1022	0.63
4.5	3	0.57	7	0.59	14	0.56	27	0.6	196	0.61	358	0.65	636	0.64	949	0.62
5	5	0.58	12	0.55	20	0.59	30	0.55	166	0.6	455	0.64	867	0.64	1128	0.62
5.5	6	0.57	12	0.55	25	0.57	30	0.6	196	0.58	520	0.63	914	0.64	891	0.64
6	5	0.56	14	0.59	23	0.57	37	0.57	239	0.59	473	0.63	764	0.63	901	0.64
6.5	6	0.56	17	0.56	26	0.56	37	0.58	188	0.61	563	0.63	833	0.63	1130	0.63
7	6	0.58	14	0.57	24	0.6	46	0.57	217	0.61	702	0.64	833	0.64	1037	0.62
7.5	7	0.56	19	0.58	36	0.56	45	0.57	239	0.6	731	0.64	621	0.63	1100	0.62
8	7	0.53	17	0.57	30	0.56	52	0.57	268	0.6	642	0.63	767	0.62	950	0.64
8.5	8	0.57	16	0.55	32	0.58	56	0.55	292	0.6	654	0.62	759	0.64	893	0.64
9	10	0.53	15	0.52	33	0.56	54	0.58	295	0.58	626	0.63	798	0.63	1041	0.65
9.5	8	0.56	20	0.56	41	0.57	63	0.59	347	0.61	651	0.63	880	0.64	989	0.65
10	8	0.54	23	0.54	40	0.52	64	0.59	371	0.6	795	0.62	821	0.64	1192	0.62
100	59	0.52	87	0.49	116	0.54	112	0.58	479	0.57	726	0.61	706	0.63	1120	0.62

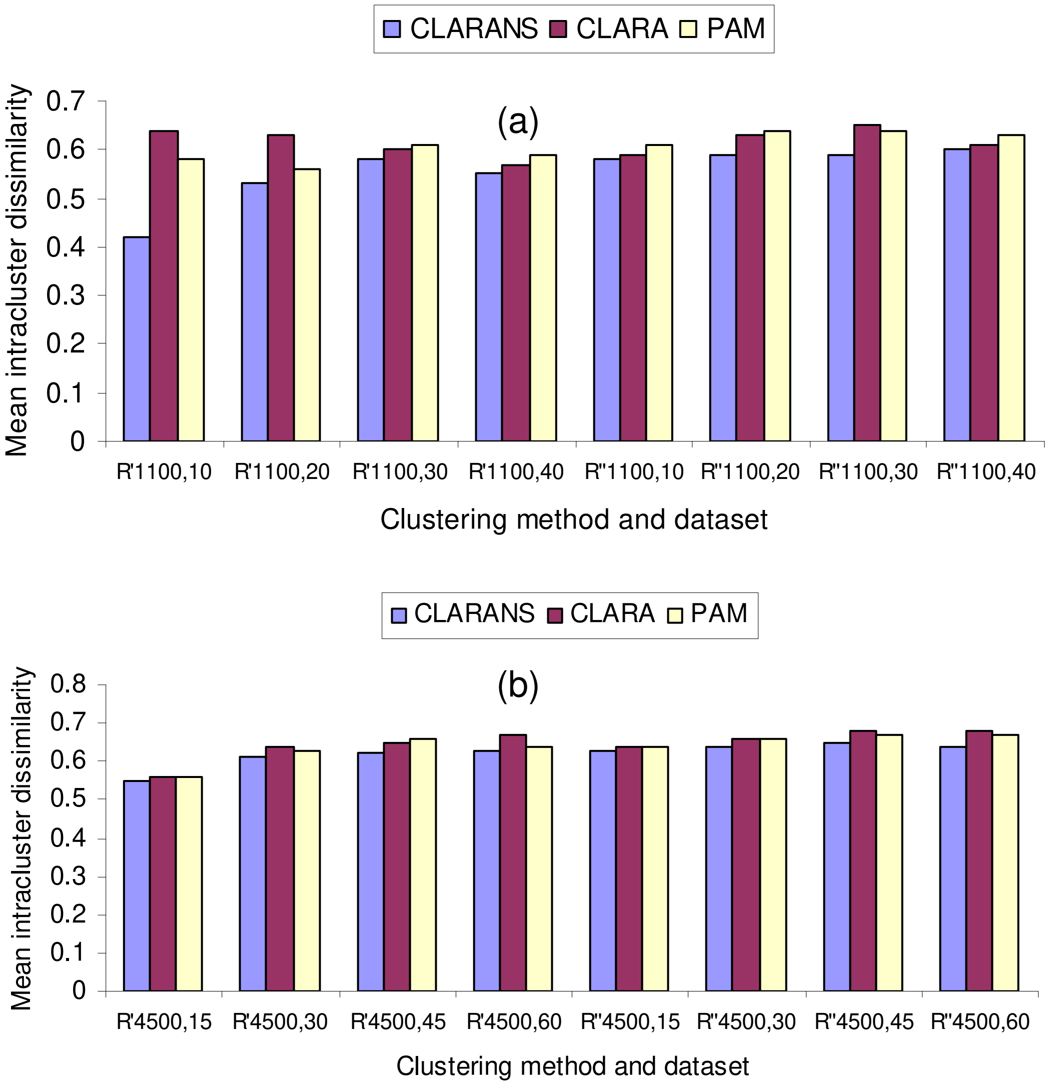


Figure 44. Comparison of mean intra-cluster similarities of CLARANS, CLARA, and PAM for: (a) datasets $R'_{1100,k}$ & $R''_{1100,k}$ and (b) $R'_{4500,k}$ & $R''_{4500,k}$.

Table 9. *Mean execution time for CLARANS, CLARA, and PAM for datasets $R_{n,k}$.*

	Execution time (s)				Execution time (s)		
Dataset	CLARANS	CLARA	PAM	Dataset	CLARANS	CLARA	PAM
$R'_{1100,10}$	1	0.15	84	$R''_{1100,10}$	1	0.14	84
$R'_{1100,20}$	2	0.5	169	$R''_{1100,20}$	2	0.5	169
$R'_{1100,30}$	6	1.31	255	$R''_{1100,30}$	5	1.28	255
$R'_{1100,40}$	6	2.58	343	$R''_{1100,40}$	9	2.65	341
$R'_{4500,15}$	45	0.35	~2.3hr	$R''_{4500,15}$	43	0.37	~2.3hr
$R'_{4500,30}$	148	1.48	~5.3hr	$R''_{4500,30}$	137	1.47	~5.3hr
$R'_{4500,45}$	241	4.1	~7 hr	$R''_{4500,45}$	203	8.87	~7 hr
$R'_{4500,60}$	397	8.95	~9.3hr	$R''_{4500,60}$	307	8.44	~9.3hr

5.6. Evaluation of Collaborative Filtering-based Prediction Models

This section presents an evaluation of the effectiveness of prediction models for user interests, using collaborative filtering techniques applied to the mined association rules. As discussed in Section 4.5, the evaluation data were the association rules matrix derived from the individual association rules that were mined from the user access logs.

As discussed in Section 4.5, two collaborative filtering models were studied: Models I used the center-based approach to compute neighborhood vectors, and Model II used the aggregate-based approach. Model II was found to be very expensive in terms of computation time and accuracy, while the results of Model I were promising.

We recall that prediction of a current user's interest involved the following steps:

- Determination of nearest neighboring navigation vectors
- Computation of a recommendation score for each of the pages of the web site using either (1) a simple average of the scores of the neighboring vector ratings, or (2) a weighted average, which computed the average of the neighboring vector scores, weighted by the cosine similarity between the current user's navigation vector and the neighboring navigation vector

Evaluation of the predictions involved setting different cosine similarity thresholds for a navigation behavior vector to belong to the current neighborhood, computing the mean prediction error for all the test data vectors, and selecting the similarity threshold that resulted in the smallest error.

5.6.1. Collaborative Filtering Prediction Model I

The distribution of mean absolute errors for different thresholds of the cosine similarity between the active user's navigation vector and the nearest neighbor navigation vectors for Collaborative Filtering Model I using average confidence values to compute predictions are shown in Figure 45 (a); Figure 45 (b) shows the distribution of the corresponding standard deviations of error. Figure 46 (a) and (b) show similar distributions when weighted average confidence values were used to compute the predictions.

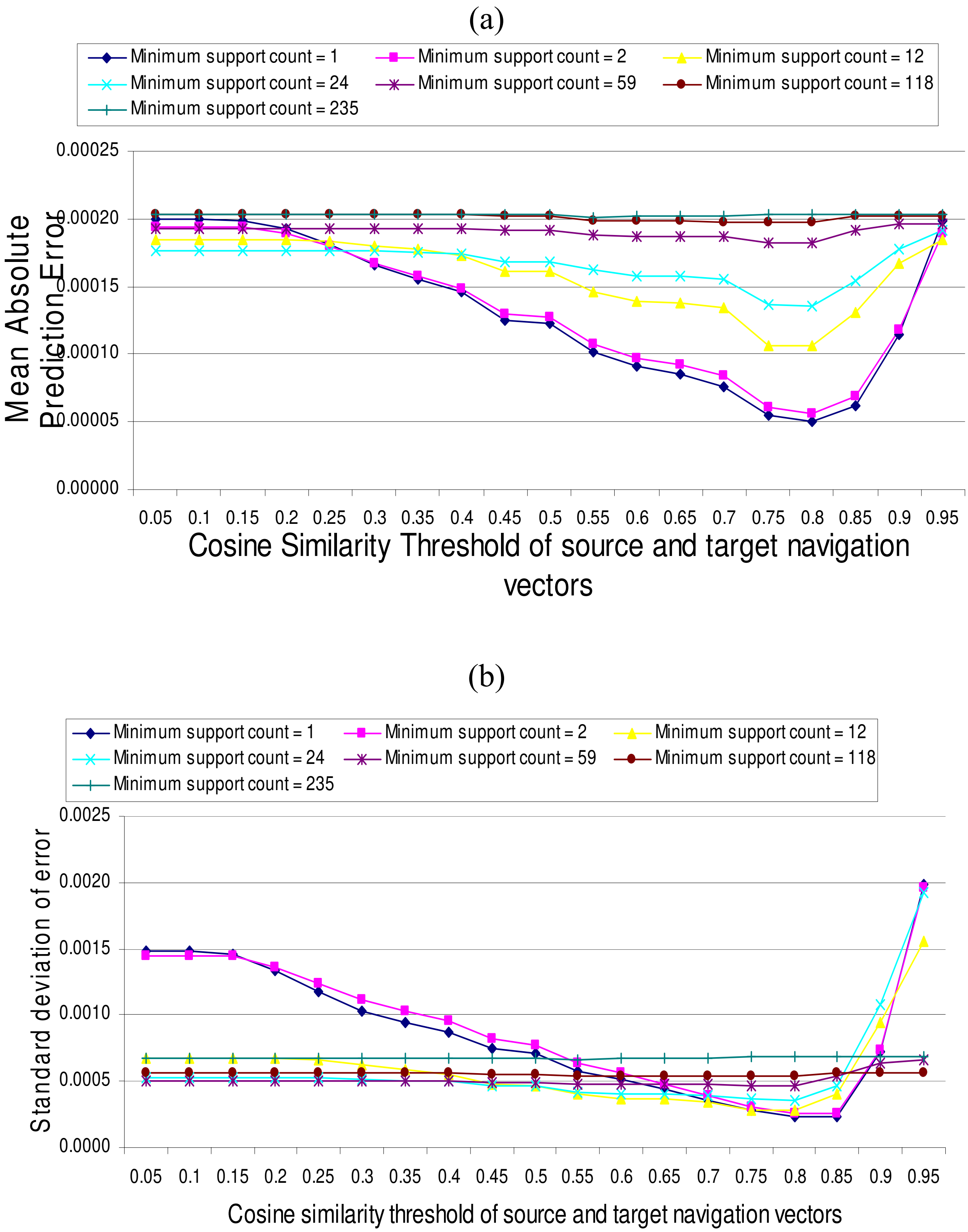


Figure 45. (a) Mean absolute errors on prediction of user interests vs. cosine similarity thresholds for various values of association rule minimum support using the center-based neighborhood formation scheme for predictions computed using average of neighborhood vectors; (b) corresponding standard deviations of error.

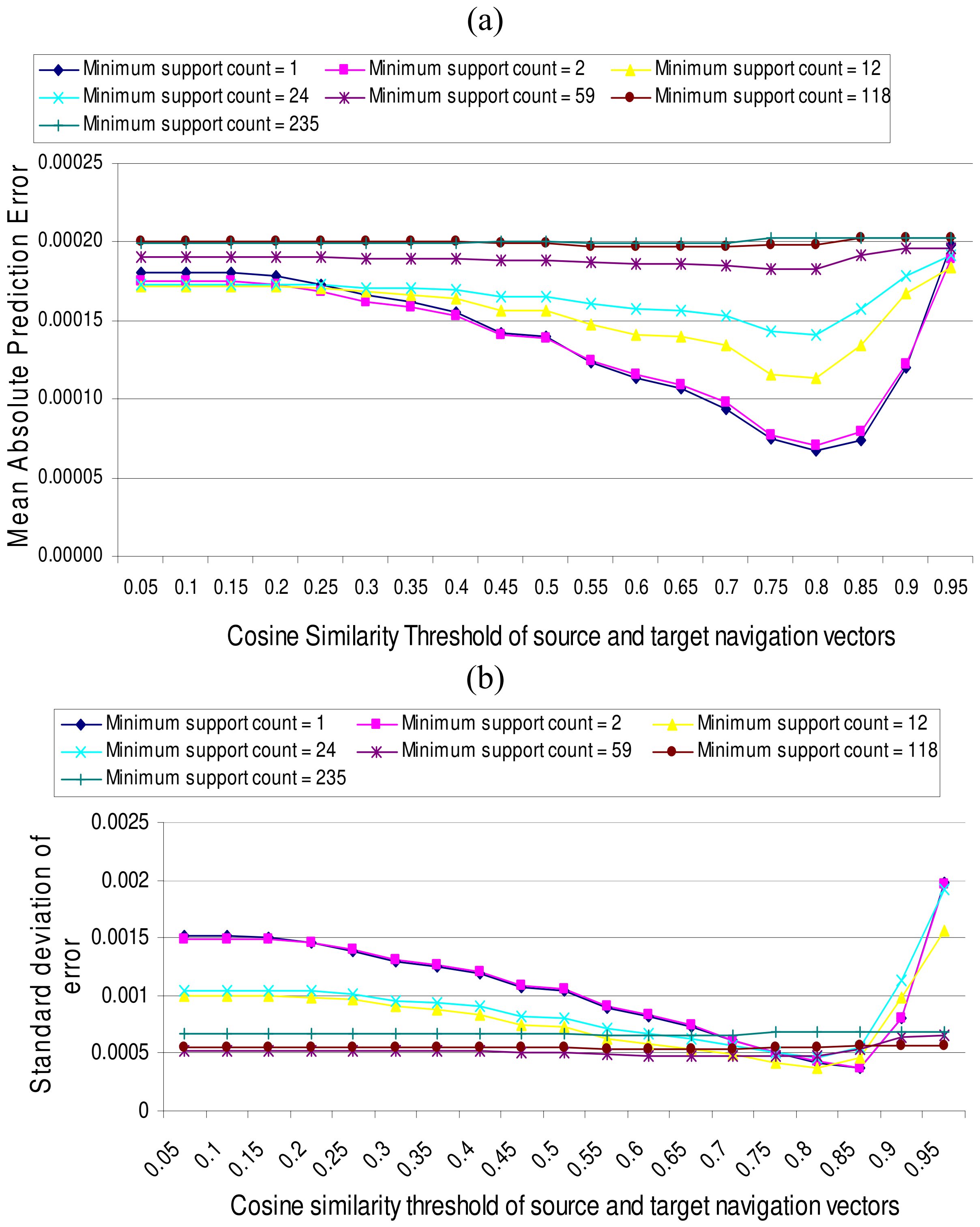


Figure 46. (a) Mean absolute errors on prediction of user interests vs. cosine similarity thresholds for various values of association rule minimum support using the center-based neighborhood formation scheme for predictions computed using weighted average of neighborhood vectors; (b) corresponding standard deviations of error.

The variation in prediction errors is similar in both cases, and had the following characteristics:

- High prediction errors for small and large values of cosine similarity threshold, and a dip in the prediction errors for cosine similarity thresholds between these extremes
- Prediction error consistently lower for association rule matrices computed from association rules with low minimum support counts than from rules with high minimum support count
- Minimum mean prediction error occurred in both cases when the cosine similarity threshold was 0.8

The standard deviations of mean prediction error were fairly constant for small and large values of cosine similarity threshold, but decreased from high values to a minimum at the 0.8 cosine similarity threshold for the same association rules (those with low minimum support count) that resulted in low prediction errors. This was highly desirable.

To summarize, the optimal prediction models using the center-based neighborhood scheme had the following properties:

- Association rules were generated with a minimum support count of 1
- Only neighborhood vectors with a cosine similarity threshold equal to or greater that 0.8 were used in the computation of predictions of user interests

Comparison of Predictions Computed Using Average and Weighted Average Values for Collaborative Filtering Model I

Consider a model that predicts test (or target) data from training data. Assuming that the data can be classified as low, medium and high, Figure 47 illustrates the different predictions that the model can make for given target values. The ideal model would predict low values if the target value were low, medium values for medium target values, and high values for high target values (i.e., the shaded regions in the diagram). In the following paragraphs, we compare the predictions computed using the average and weighted average confidence values by comparing the results of the models to Figure 47.

		Predicted Score		
		Low	Medium	High
Target Score	Low	DESIRED	Prediction HIGH	Prediction TOO HIGH
	Medium	Prediction LOW	DESIRED	Prediction HIGH
	High	Prediction TOO LOW	Prediction LOW	DESIRED

Figure 47. Illustrating regions of operation of a prediction model. The shaded cells show the desired region of operation. All other cells within the thick borders are labeled with the prediction errors.

Figure 48 shows the distribution of the prediction scores using training data alone, for known test (target) data using simple averages of neighborhood training data confidence scores to compute the predictions. The row and column labels of the table begin from "0", and increase in steps of 0.2, up to 1.0. The address of each cell is of the form (*target range*, *predicted range*). Hence, Cell(0, 0) contains the number of HTML pages that were predicted to have a score of "0" when the target score was "0"; Cell(0.2, 0.4) the number of pages that were predicted to have a score of between 0.2 and 0.4 when the target score was between 0 and 0.2, etc. Each row in the figure comprised the following:

- The count of the target value that is being predicted
- The target value that is being predicted
- Counts of a range of predictions for the given target values

The following procedure was followed to generate the figure:

- For each test vector, the known target value for each component was noted and the corresponding count incremented.
- The prediction vector for the given test vector was then computed, the vector components noted, and the corresponding prediction counts incremented.

Figure 49 shows the same distributions as Figure 48, but with the counts replaced by their corresponding percentages. Figure 50 and Figure 51 show similar distributions as Figure 48 and Figure 49, but this time for predictions computed using weighted average scores of the neighborhood vectors. Comparing the prediction results computed from the average and weighted average approaches, we see that the weighted average approach fits better into the model depicted in Figure 47, suggesting that the weighted average model is preferred over the simple average model. This observation was investigated further using the signal detection theory approach. For each row of Figure 48 and Figure 50, the following table was generated,

		Actual prediction	
		True	False
Target prediction	True	TT	TF
	False	FT	FF

where, for each target value $\boldsymbol{T}$

Cell TT is the is the number of predictions $\boldsymbol{P}$ that are equal to $\boldsymbol{T}$ within a predetermined threshold $\boldsymbol{h}$ (i.e. $\boldsymbol{P} = \boldsymbol{T} \pm \boldsymbol{hval} \mid 0 \le \boldsymbol{hval} \le \boldsymbol{h}$);

Cell TF is the number of predictions for $\boldsymbol{T}$ that are outside the threshold $\boldsymbol{h}$, i.e., different from $\boldsymbol{T} \pm \boldsymbol{hval}$

Cell FT is the number of non-target values $\boldsymbol{NT}$ whose predictions are within threshold $\boldsymbol{h}$, (i.e. $\boldsymbol{P} \neq \boldsymbol{NT} \pm \boldsymbol{hval} \mid 0 \le \boldsymbol{hval} \le \boldsymbol{h}$); and

Cell FF is the number of non-target values whose predictions are outside threshold $\boldsymbol{h}$, i.e., different from $\boldsymbol{T} \pm \boldsymbol{hval}$.

Count	Target	Prediction																																						
		0	.02	.04	.06	.08	.10	.12	.14	.16	.18	.20	.22	.24	.26	.28	.30	.32	.34	.36	.38	.40	.42	.44	.46	.48	.50	.52	.54	.56	.58	.60	.62	.64	.66	.68	.70	.72	.74	.76	.78	1
13703786	0	13666900	25736	4522	2013	1199	675	530	406	279	242	171	144	125	129	71	64	67	57	52	48	52	39	38	35	31	36	27	13	9	8	7	4	3	7	11	3	5	5	7	3	17
3427	.02	3099	255	21	16	8	6	2	2	1	2	2	1	2	2	1	2		2			1														1						1
1679	.04	1432	137	37	25	8	7	5	4	2	3	2	1	2	1			1	3					1			1									1						2
926	.06	717	86	56	21	10	5	4	5	3		3	2	1	2	1	2		2			1					1									1						2
629	.08	435	64	44	26	12	13	6	4	5	1	1	1	1	3	1	1		1	1		1		1			1				1					1						2
521	.1	331	39	35	27	20	14	9	12	6	3	4	1	1	1	1	1	2	1	1	3	2	1	1			2														1	1
395	.12	248	32	25	19	12	8	4	6	6	5	4		4	3	2	1	1	1	1	1	3	1		1		1	1			1					1						1
329	.14	172	25	22	25	18	15	6	6	6	5	3	2	2	2	4	2	1	1	1	1		1	1		1	1				1					1						1
325	.16	150	30	25	20	14	13	8	12	5	9	8	3	6	2	3	3		2	1	2	1		1	1	1										1		1				1
242	.18	128	16	17	8	8	6	5	9	5	6	4	1	3	4	2	3	2	2	1	2	2	1	1			3						1	1								1
336	.2	135	30	14	14	8	12	14	9	9	16	22	4	5	10	2	4	4	1	3	1	4	1	3	1		3			1			1			1						2
116	.22	60	8	3	4	1	2	3	5	4	4	2	2	2	1	2	2	1	1	2	1	1		1						1												1
394	.24	203	20	10	10	10	8	7	9	15	20	15	11	12	9	5	4	4	2	1	1	3	1	3	2	1	3		1	1	1		1			1						
256	.26	87	18	10	10	9	8	6	3	8	10	14	20	10	6	5	8	3	4	3	2	3		1	1	1	1		2	1		1	1							1		
125	.28	49	5	4	1	2	5	2	5	4	7	3	3	4	3	1	4	3	2	2		2	1	1	1	1	1				1		1			1				1		1
496	.3	86	25	13	18	17	15	14	18	32	27	30	37	26	24	20	32	10	13	3	2	4	4	5	2	2	6	1	1	1		2	1			2	1		1	1		1
81	.32	28	4	2	3	2	3		6		3	2	1	5	2	2	2	2	1	2	1	1			1	2	2				1					1						
363	.34	100	22	15	12	10	6	9	10	13	16	9	9	11	5	14	11	20	18	6	6	8	6	3	2	2	4	2	3	3		1	1	1		4						3
94	.36	22	3	3	4	3	3	1	1		3	1	3	1	2	2	1	3	4	5	6	3	4	3	2	3	1	2	1	1	1	1			1			1				
125	.38	34	4	6	6	5	2	4	3	7	6	6	11	3	2	2	1	2	2	1	1	1	2	2	1	1	3	1	1			1						1				2
282	.4	89	18	15	10	10	7	6	8	6	7	9	6	7	7	6	4	5	7	5	2	9	2	4	3	3	6	2	2	4	1	3	1	1	1	5			1			1
33	.42	14		1	1		1		1	1		1			1				1				1			1	1	1		1				1		1			1			1
102	.44	25	4	6	2	1	2	2		1	4	3	3	3	2	3		3	1	2	1	4	2	2	1	1	5	2	1	1	2	2	1	1	1	2		1		1		3
82	.46	26	5	4	2	3	1		1	2	2	2	1		3	3	1	2	1		1	3	1	1	1	1	2		1	1	1	1	2	1		1	1	1		1	1	1
35	.48	9	2	3	1				1		1	1		1	1	1	1	1	1			1	1			2	1		1							1						
826	.5	380	30	18	13	12	13	7	11	7	10	6	6	10	12	11	13	16	13	14	18	16	12	16	23	22	27	12	10	6	2	7	6	4	5	8	6	3	2	6	3	8
6	.52	3						1		1																																
26	.54	11						2										1				1				2	1			1	2			1			1	1	1			
89	.56	15	2		2	1	2		1	1	1	1	2	2	2	4	2	3	1	2		1		1	1		4	2		3	3	4	1	6	4	3	2	1	3	2		1
79	.58	19	3	4	2	4	1	1	3			1		1		1	2	2	1	1	1	1	2	2	1	2	1	2	2		1	3	1		1	4	1	1	1	2	1	3
186	.6	51	9	6	4	3	3	1	1	3	2	1	1	4	2	1	4	3	2	2	1	1	5	1	2	2	4	2	3	5	3	23	1	1	1	3	3	1	1	2	4	10
18	.62	6			1		1	1							1						1										1				1				1			1
109	.64	13	2	2	3	1		1	1	1	1	3	2	1	3	2	2	2	7	5	6	4	2		1		1		1	1	1	3	7	2	2	5	5	6	1			6
11	.66	2										1										1			1					1									1	1		
668	.68	285	31	15	14	9	8	8	10	6	11	8	7	6	6	13	13	13	10	4	8	14	8	8	4	5	14	6	3	8	7	7	10	6	9	35	4	11	2	5	2	16
16	.7	5		1	1	1	1												1																							2
184	.72	21	2	2	2	2	3	1	2	1		2	2	2		2		1	1	2	1	3	1	2			2	1	1	1		2	1	8	3	4	13	11	7	25	10	39
20	.74	5				1																2										1							2	1	1	3
304	.76	77	5	17	4	4	2	2	2		2	2	2	1	2	4	1	4	2	3	3	2	3	2		2	4	2	2	2	2	5	4	3	3	7	1	8	7	14	17	73
146	.78	116			1	1	1		1		1		1			1	1	1				1		1		1		1			1			1			1	1	2		1	8
3494	1	270	29	44	25	19	20	8	9	14	9	10	10	8	11	23	7	19	12	24	17	16	23	23	30	34	27	23	18	13	17	17	20	20	24	25	24	49	41	46	63	2352

Figure 48. Distribution of counts of target and prediction scores for Prediction Model I, based on average neighborhood scores.

	Prediction																																								
Target	0	.02	.04	.06	.08	.10	.12	.14	.16	.18	.20	.22	.24	.26	.28	.30	.32	.34	.36	.38	.40	.42	.44	.46	.48	.50	.52	.54	.56	.58	.60	.62	.64	.66	.68	.70	.72	.74	.76	.78	1
0	1																																								
.02	.90	.07	.01																																						
.04	.85	.08	.02	.01																																					
.06	.77	.09	.06	.02	.01	.01		.01																																	
.08	.69	.10	.07	.04	.02	.02	.01	.01	.01																																
.10	.64	.08	.07	.05	.04	.03	.02	.02	.01	.01	.01									.01																					
.12	.63	.08	.06	.05	.03	.02	.01	.02	.02	.01	.01		.01	.01	.01						.01																				
.14	.52	.08	.07	.08	.05	.05	.02	.02	.02	.01	.01	.01	.01	.01	.01	.01																									
.16	.46	.09	.08	.06	.04	.04	.02	.04	.02	.03	.03	.01	.02	.01	.01	.01		.01		.01																					
.18	.53	.07	.07	.03	.03	.02	.02	.04	.02	.03	.02		.01	.02	.01	.01	.01	.01	.01	.01	.01					.01															
.20	.40	.09	.04	.04	.02	.03	.04	.03	.03	.05	.07	.01	.01	.03	.01	.01	.01		.01		.01		.01			.01															.01
.22	.52	.07	.03	.04	.01	.01	.03	.04	.03	.03	.02	.01	.02	.01	.01	.02	.01	.01	.02	.01	.01		.01						.01												.01
.24	.51	.05	.03	.02	.03	.02	.02	.02	.04	.05	.04	.03	.03	.02	.01	.01	.01	.01			.01		.01			.01															
.26	.34	.07	.04	.04	.04	.03	.02	.01	.03	.04	.05	.08	.04	.02	.02	.03	.01	.02	.01	.01	.01		.01					.01													
.28	.39	.04	.03	.01	.01	.04	.02	.04	.03	.05	.03	.03	.03	.02	.01	.03	.02	.01	.02		.02	.01	.01	.01	.01	.01				.01		.01			.01				.01		.01
.30	.17	.05	.03	.04	.03	.03	.03	.04	.07	.05	.06	.08	.05	.05	.04	.07	.02	.03	.01		.01	.01	.01			.01															
.32	.35	.05	.02	.04	.02	.03		.07		.03	.03	.01	.06	.03	.02	.03	.03	.02	.02	.02	.01			.02	.02	.02				.01					.01						
.34	.27	.06	.04	.03	.03	.02	.02	.03	.03	.04	.03	.02	.03	.01	.04	.03	.06	.05	.02	.02	.02	.02	.01	.01		.01	.01	.01	.01						.01						.01
.36	.24	.03	.03	.04	.03	.03	.01	.01		.03	.01	.04	.01	.02	.02	.01	.03	.05	.05	.06	.03	.04	.03	.02	.03	.01	.02	.01	.01	.01	.01			.01			.01				
.38	.27	.03	.05	.05	.04	.02	.03	.03	.06	.05	.05	.09	.02	.01	.02	.01	.02	.02	.01	.01	.01	.02	.01	.01	.01	.02	.01	.01			.01						.01				.01
.40	.32	.06	.05	.04	.03	.02	.02	.03	.02	.02	.03	.02	.03	.02	.02	.01	.02	.03	.02	.01	.03	.01	.01	.01	.01	.02	.01	.01	.01		.01				.02						
.42	.42	.01	.04	.03	.01	.03		.02	.02		.02		.01	.03	.01			.03				.03	.01	.01	.02	.02	.03		.03	.01	.01	.01	.02	.01	.03	.01	.01	.03			.02
.44	.25	.04	.06	.02	.01	.02	.02		.01	.04	.03	.03	.03	.02	.03		.03	.01	.02	.01	.04	.02	.02	.01	.01	.05	.02	.01	.01	.02	.02	.01	.01	.01	.02		.01		.01		.03
.46	.31	.06	.04	.02	.04	.02		.01	.02	.03	.02	.02		.03	.04	.01	.02	.01		.01	.03	.02	.01	.02	.01	.03		.01	.01	.02	.01	.02	.02		.01	.01	.01		.01	.01	.01
.48	.26	.05	.08	.04		.01	.01	.02	.01	.03	.02	.01	.02	.04	.03	.03	.04	.04	.01	.01	.04	.03	.01	.01	.07	.03		.03	.01						.02		.01		.01		.01
.50	.46	.04	.02	.02	.01	.02	.01	.01	.01	.01	.01	.01	.01	.01	.01	.02	.02	.02	.02	.02	.02	.01	.02	.03	.03	.03	.01	.01	.01		.01	.01		.01	.01	.01			.01		.01
.52	.42		.05				.16	.05	.11	.05	.05									.05															.05						
.54	.41			.01	.01		.06			.01	.01		.01	.01	.01		.03		.01	.01	.03		.01	.01	.06	.03			.03	.06	.01		.04			.03	.03	.05			
.56	.17	.03		.02	.01	.03		.01	.01	.01	.01	.02	.02	.03	.04	.02	.04	.01	.02		.01		.01	.01		.04	.02		.03	.03	.05	.01	.07	.04	.04	.02	.01	.03	.02		.01
.58	.24	.04	.05	.03	.05	.02	.02	.03			.01		.01		.01	.02	.03	.01	.01	.02	.01	.02	.02	.01	.03	.01	.02	.03		.02	.04	.01		.01	.06	.02	.01	.01	.03	.02	.04
.60	.28	.05	.03	.02	.02	.02	.01	.01	.02	.01		.01	.02	.01	.01	.02	.02	.01	.01	.01	.01	.03	.01	.01	.01	.02	.01	.02	.03	.02	.13	.01	.01	.01	.02	.02	.01	.01	.01	.02	.05
.62	.36	.02		.04		.04	.04				.02		.02	.04		.02	.02			.04	.02	.02	.02	.02		.02			.02	.04				.04		.02		.08	.02		.06
.64	.12	.02	.02	.03	.01		.01	.01	.01	.01	.03	.02	.01	.02	.02	.02	.02	.07	.04	.06	.03	.02		.01		.01		.01	.01	.01	.03	.06	.02	.02	.05	.05	.05	.01			.06
.66	.18	.03	.03		.03	.03					.09		.03	.03	.03			.03			.06			.06					.12				.03		.03			.06	.06	.03	.03
.68	.43	.05	.02	.02	.01	.01	.01	.01	.01	.02	.01	.01	.01	.01	.02	.02	.02	.01	.01	.01	.02	.01	.01	.01	.01	.02	.01		.01	.01	.01	.02	.01	.01	.05	.01	.02		.01		.02
.70	.29		.06	.04	.04	.04	.02	.02	.02	.02		.02		.02	.02		.02	.04		.02	.02			.02	.02		.02				.02					.02	.02		.02	.02	.12
.72	.12	.01	.01	.01	.01	.01	.01	.01			.01	.01	.01		.01		.01	.01	.01	.01	.01	.01	.01			.01	.01	.01			.01		.04	.02	.02	.07	.06	.04	.14	.06	.21
.74	.25	.02	.02	.02	.05	.02				.02	.02			.02		.02	.02			.02	.08				.02	.02			.02	.02	.03						.02	.08	.05	.03	.17
.76	.25	.02	.05	.01	.01	.01	.01	.01		.01	.01	.01		.01	.01		.01	.01	.01	.01	.01	.01	.01		.01	.01	.01	.01	.01	.01	.02	.01	.01	.01	.02		.03	.02	.05	.06	.24
.78	.79				.01			.01		.01						.01					.01				.01					.01							.01	.01		.01	.06
1.00	.08	.01	.01	.01	.01	.01									.01		.01		.01			.01	.01	.01	.01	.01	.01	.01				.01	.01	.01	.01	.01	.01	.01	.01	.02	.67

Figure 49. Distribution of percentages of target and prediction scores for Prediction Model I, based on average neighborhood scores.

		Prediction																																								
Count	Target	0	.02	.04	.06	.08	.10	.12	.14	.16	.18	.20	.22	.24	.26	.28	.30	.32	.34	.36	.38	.40	.42	.44	.46	.48	.50	.52	.54	.56	.58	.60	.62	.64	.66	.68	.70	.72	.74	.76	.78	1
18544949	0	18531007	9469	2140	992	495	273	189	117	84	59	34	29	21	10	7	7	6	3	1	1	1	1		1																	
3520	.02	3367	129	15	5	2	2																																			
1677	.04	1490	122	30	10	9	7	4	2	1			1																													
923	.06	747	78	39	11	13	11	5	6	4	3	2		1	1	1																										
623	.08	439	39	51	23	17	8	6	8	9	6	5	4	1		2			1				1																			1
507	.1	319	21	41	31	18	18	16	11	5	4	2	1	3	2	4	2		1	1	2	1	2																			1
378	.12	225	8	26	24	17	10	12	13	6	7	5	6	1	1	3	2	1	1	1	1	1	1		1	1						1										1
336	.14	166	3	19	23	23	20	18	15	5	11	5	3	5	2	3	1	3	1	1	3	1	2	1						3												1
330	.16	165	5	14	20	15	16	19	16	11	11	6	5	4	3	5	2	2	2	1	1		2			1				1			1									1
248	.18	125		5	12	8	14	14	10	9	9	6	8	5	3	5	2	1	2	2	1	1	4		1																	1
362	.2	151	2	6	6	14	17	21	21	20	26	11	13	5	5	6	4	2	5	3	2	1	4	1	1	1	1	2		4	1	1	2									3
112	.22	55	1	3	2	2	2	3	5	6	3	4	4	2	2	4	3	3	2	1	1		1			1				1	1											
385	.24	96		4	5	5	9	10	19	51	88	22	12	9	8	8	3	2	4	2	3	3	6	1	1	1	1	2		4	1			1								2
263	.26	84		3	4	6	6	12	12	22	24	11	15	10	10	6	2	3	4	4	2	2	6	2	2	1	1		1	4			1									3
134	.28	46			1	1	3	3	5	10	9	7	5	7	7	4	2	4	1	2	1	2	4	1	1	1	1			1	1											1
483	.3	83		1	2	5	6	16	21	52	62	58	31	20	38	21	7	3	8	4	4	2	13	4	3	2	2		1	4	1	1	2	1								3
84	.32	24				1	1	1	2	2	4	4	4	3	4	5	6	2	6	3	1	1	1			1	1	1		1			1									
391	.34	100		2	3	2	5	6	10	9	17	20	26	22	17	28	17	12	13	9	8	4	9	7	2	5	2	2	2	8	5	5	3	2	1		1					8
85	.36	11				1	1	1	1	2	4	2	2	1	2	4	6	11	9	7	5	2	1	1	2	2	1			1	2											
139	.38	31			1	1	1	3	1	4	5	8	12	13	5	7	5	3	5	2	4	2	7	1	2	1	1	2	1	3	2		2		1		1					3
280	.4	69			1	2	3	3	2	5	8	8	12	8	16	10	9	7	13	8	8	5	14	8	8	4	5	5	4	11	2	2	4	1	1	1	5					9
33	.42	13					1							1	1	1	1	1			1	1		2	1	2				2	2	1										
101	.44	24						1	1	2	2	2	5	1	7	4	2	3	6	2	4	1	6	3	2	3	3	3	1	3		1	2			1	1					2
86	.46	20									1	2	4	4	2	3	2	3	2	3	4	3	3	2	1	2	3	5	3	2	2	1	2	2			1					2
39	.48	11										1		2	2	3	2		2	1	3	2	3		1	1		1				1										
806	.5	107		1	1	3	2	3	3	2	5	8	13	10	13	15	16	19	24	37	77	50	66	42	34	55	48	26	19	29	12	10	15	6	4	6	8	2	3	1	1	14
6	.52	4																					1		1																	
24	.54	5														1	1	1	1	1	1	1		1	1	1	1	2	3	1		1	1								
92	.56	11										1	1	2	1	4	3	4		1		2	1	1	3	3	7	9	10	9	7	6	2	1		1					2
70	.58	18							1	1			1	1	1		1	1	2	1	1	2	1	1	4	3	3	4	2	1	3	6	4	2	1	1	1	1		1	2
181	.6	33									1		1	1	3	1	3	2	2	3	1	6	4	4	8	11	35	8	10	9	5	7	3	5	3	3	1	1	1		6
19	.62	7																			1	1	1				1		2	1			2				1				
110	.64	12											1		2	1	1	2	1	1	2	2		2	2	2	6	12	14	10	11	5	7	3	1	1	1	1	1	2	3
11	.66	3											1		1		1			1	1	1				1			1					1							
672	.68	88					2	2	2	1		3	2	4	5	6	6	7	11	10	10	7	12	12	14	14	18	22	15	51	32	44	60	53	41	20	23	13	17	10	5	33
18	.7	5														1						1		1		1	1		1	1		1		1		1		1			1
197	.72	14											1		1			1			1	2	1	1	1	2	3	4	10	16	19	20	36	30	14	8	3	2	1	1	3
26	.74	6																							1	1	1	1	2	1	2	4	1	1		1	1				1
270	.76	24																1	1	2	1	1	2	3	2	3	6	3	10	9	6	19	19	34	23	14	17	24	20	11	14
143	.78	6													1	1								2	1		1	3	1	1	3	1	5	13	55	37	7	1	1		3
6096	1	242				1	2			1		1	1	2	2	3	4	2	4	4	7	5	12	8	14	17	16	20	19	25	34	49	58	67	98	101	149	150	187	209	347	4233

Figure 50. Distribution of counts of target and prediction scores for Prediction Model I, based on weighted average neighborhood scores.

	Prediction																																								
Target	0	.02	.04	.06	.08	.10	.12	.14	.16	.18	.20	.22	.24	.26	.28	.30	.32	.34	.36	.38	.40	.42	.44	.46	.48	.50	.52	.54	.56	.58	.60	.62	.64	.66	.68	.70	.72	.74	.76	.78	1.00
0	1.00																																								
.02	.96	.04																																							
.04	.89	.07	.02	.01	.01																																				
.06	.81	.08	.04	.01	.01	.01	.01	.01																																	
.08	.70	.06	.08	.04	.03	.01	.01	.01	.01	.01	.01	.01																													
.10	.63	.04	.08	.06	.04	.04	.03	.02	.01	.01			.01		.01																										
.12	.60	.02	.07	.06	.05	.03	.03	.03	.02	.02	.01	.02			.01	.01																									
.14	.49	.01	.06	.07	.07	.06	.05	.04	.01	.03	.01	.01	.01	.01	.01		.01			.01		.01								.01											
.16	.50	.02	.04	.06	.05	.05	.06	.05	.03	.03	.02	.02	.01	.01	.02	.01	.01	.01				.01																			
.18	.51		.02	.05	.03	.06	.06	.04	.03	.03	.02	.03	.02	.01	.02	.01		.01	.01			.01																			
.20	.42		.02	.02	.04	.05	.06	.06	.05	.07	.03	.04	.01	.01	.02	.01	.01	.01	.01	.01		.01								.01												.01
.22	.49	.01	.03	.02	.02	.01	.03	.04	.05	.02	.03	.04	.02	.01	.04	.02	.03	.02	.01	.01		.01			.01				.01	.01											
.24	.25		.01	.01	.01	.02	.03	.05	.13	.23	.06	.03	.02	.02	.02	.01		.01	.01	.01	.01	.02							.01												.01
.26	.32		.01	.02	.02	.02	.04	.05	.08	.09	.04	.06	.04	.04	.02	.01	.01	.02	.02	.01	.01	.02	.01	.01	.01				.02												.01
.28	.34					.02	.02	.04	.08	.06	.05	.04	.05	.05	.03	.02	.03		.02	.01	.01	.03	.01		.01	.01				.01											
.30	.17				.01	.01	.03	.04	.11	.13	.12	.06	.04	.08	.04	.01	.01	.02	.01	.01		.03	.01	.01					.01												.01
.32	.29				.01	.01	.01	.03	.03	.05	.05	.05	.04	.05	.06	.07	.03	.08	.04	.01	.01	.02			.01	.01	.01		.01			.01									
.34	.26		.01	.01	.01	.01	.01	.02	.02	.04	.05	.07	.06	.04	.07	.04	.03	.03	.02	.02	.01	.02	.02		.01		.01	.01	.02	.01	.01	.01									.02
.36	.13				.01	.02	.02	.01	.02	.04	.02	.02	.01	.02	.05	.07	.13	.11	.08	.06	.03	.02	.02	.02	.03	.01			.01	.03											
.38	.22				.01	.01	.02	.01	.03	.04	.06	.08	.09	.04	.05	.04	.02	.03	.02	.03	.01	.05	.01	.02			.01	.01	.02	.01		.01									.02
.40	.25				.01	.01	.01	.01	.02	.03	.03	.04	.03	.06	.03	.03	.03	.05	.03	.03	.02	.05	.03	.03	.02	.02	.02	.01	.04	.01	.01	.02				.02					.03
.42	.39					.03		.01					.03	.03	.03	.03	.02			.04	.03	.01	.06	.03	.06	.01	.01	.01	.05	.05	.04					.01					.01
.44	.24						.01	.01	.02	.02	.02	.05	.01	.07	.04	.02	.03	.06	.02	.04	.01	.06	.03	.02	.03	.03	.03	.01	.03		.01	.02			.01	.01					.02
.46	.24									.02	.03	.05	.05	.02	.03	.03	.04	.03	.04	.05	.03	.04	.02	.02	.03	.03	.06	.04	.02	.02	.01	.02	.02			.01					.03
.48	.28					.01	.01		.01		.02	.01	.04	.04	.08	.05	.01	.06	.02	.09	.06	.08	.01	.03	.03		.03				.03		.01	.01							
.50	.13									.01	.01	.02	.01	.02	.02	.02	.02	.03	.05	.10	.06	.08	.05	.04	.07	.06	.03	.02	.04	.01	.01	.02	.01		.01	.01					.02
.52	.58															.05			.05			.11	.05	.11	.05																
.54	.19									.01	.01	.01			.01	.03	.03	.06	.03	.03	.03	.06		.04	.04	.04	.06	.07	.11	.04		.04	.03			.01		.01			
.56	.12											.01	.01	.02	.01	.04	.03	.04		.01		.03	.01	.01	.03	.03	.08	.09	.11	.10	.08	.07	.02	.01		.01					.02
.58	.25								.01	.01			.01	.01	.01		.01	.02	.03	.01	.02	.03	.01	.01	.06	.04	.04	.06	.02	.02	.04	.08	.06	.02	.01	.01	.01	.01		.01	.02
.60	.18														.02	.01	.01	.01	.01	.02	.01	.03	.02	.02	.04	.06	.19	.04	.06	.05	.03	.04	.02	.03	.01	.02	.01				.03
.62	.37									.02						.02		.02	.02	.02	.04	.07	.04	.02	.02	.02	.05	.02	.11	.04			.09				.04				.02
.64	.11												.01		.02	.01	.01	.02	.01	.01	.02	.02		.02	.02	.02	.06	.11	.13	.09	.10	.05	.06	.02	.01	.01	.01	.01	.01	.02	.02
.66	.24												.06	.03	.06		.06			.06	.06	.06				.06	.03	.03	.06		.03			.12	.03						
.68	.13												.01	.01	.01	.01	.01	.02	.01	.01	.01	.02	.02	.02	.02	.03	.03	.02	.08	.05	.06	.09	.08	.06	.03	.03	.02	.02	.01	.01	.05
.70	.30										.02					.04		.02	.02			.04	.02	.06	.02	.04	.06	.02	.04	.04		.06		.06		.04	.02	.04			.08
.72	.07														.01			.01				.01			.01	.01	.01	.02	.05	.08	.09	.10	.18	.15	.07	.04	.02	.01	.01	.01	.02
.74	.22												.01			.01				.01			.01	.01	.04	.03	.05	.04	.06	.04	.08	.15	.04	.05	.01	.04	.03	.01	.01	.01	.03
.76	.09																			.01			.01	.01	.01	.01	.02	.01	.04	.03	.02	.07	.07	.13	.08	.05	.06	.09	.07	.04	.05
.78	.04																							.01	.01			.02	.01		.02	.01	.03	.09	.39	.26	.05	.01			.02
1.00	.04																													.01	.01	.01	.01	.02	.02	.02	.02	.03	.03	.06	.69

Figure 51. Distribution of percentages of target and prediction scores for Prediction Model I, based on weighted average neighborhood scores.

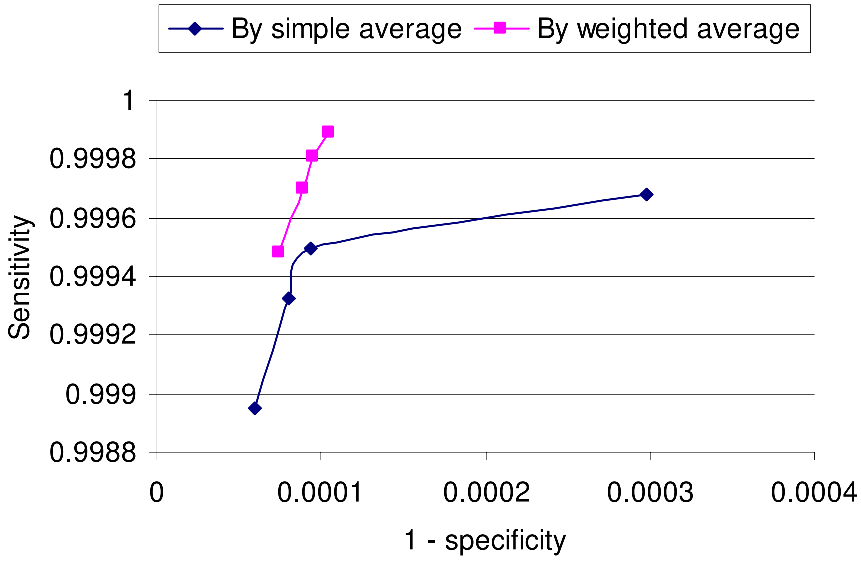


Figure 52. ROC curves for predictors of user interests based on average and weighted averages

Corresponding cells of the tables were then summed, and sensitivity and specificity derived from them as follows:

- Sensitivity = TT/(TT + TF)
- Specificity = FF/(FF + FT)

Sensitivity and specificity were determined for 4 levels of threshold ***h***: 0.04, 0.08, 0.12, and 0.16. The resulting ROC curves are shown in Figure 52. As can be seen from the figure, the curve corresponding to the weighted average predictor is closer to the left and upper borders, confirming our suspicion that it is a better predictor than the predictor based on simple average of neighboring vector values.

5.6.2. Collaborative Filtering Prediction Model II

As was mentioned in Section 4.5, Prediction Model II computed user interests by using contributions for neighborhood vectors generated user the aggregate-based scheme. Preliminary investigation of the model showed that it was computationally too expensive, even after the following simplifications were made:

- A limit of 50 nearest neighboring vectors in a neighborhood
- Selection of a maximum of 250 items from the target data for the computation of predictions.

Table 10 for instance, shows the computation times for 3 iterations of the algorithm, using association rule matrices computed with various minimum support counts.

Apart from the large computation times involved, accuracy of this model was also a problem, with all predictions equal to, or close to zero. Because of these difficulties, this model was not pursued any further.

Table 10. *Association rule counts and computation times for 3 iterations of program used to compute predictions using aggregate-based scheme for determination of KNN vectors.*

	Minimum rule support count						
	1	2	3	4	5	6	7
Rule count	99426	86477	50120	41046	33403	26285	16293
Computation time (min)	1172	814	126	60	40	36	25

6. CONCLUSIONS AND FUTURE DIRECTIONS

As mentioned in the Introduction, the objective of this research was to examine the design process of hyperlink recommender systems that deduce user profiles by comparing their navigation behaviors to those of past users of a web site, and propose improvements. This section presents the following: a summary of the approach used – its strengths, weaknesses, and possible improvements; a summary of findings of the research; and pointers to future directions of the work.

Approach. The underlying assumption of the work was that if users of a web site exhibit similar navigation behaviors, then they must be interested in the same set of web pages. User navigation behaviors were deduced by applying heuristics to determine user sessions from user access logs, and transactions (or user navigation behaviors) within sessions. The transactions database was then mined to discover useful user navigation behaviors, which were used as data in a collaborative filtering model for predicting the current user's interests.

Determination of user navigation behaviors as described above is not new (see for example, Pirolli et al. (1996), Pitkow (1997), and Chen (1996) for heuristics used in the determination of user navigation behaviors. What is new is the way these navigation behaviors have been analyzed to predict user interests. In work by and colleagues (Cooley et al., 1997, 1999), prediction was done by matching a window corresponding to the last *n* pages visited by the user to identical patterns mined from server logs, and performing a simple computation to determine prediction score. In this research, we have harnessed the power of collaborative filtering to consider not only identical navigation patterns in the computation of prediction score, but also similar navigation patterns, provided the similarity of these patterns to the current user's navigation vector exceeds a threshold.

The main advantages of using user access logs are their plentiful supply, which leads to a better chance of discovering meaningful patterns within the data, and the fact that the user is spared the burden of having to specify their profile to the system. The main disadvantages of this approach stem from the noisy nature of the logs, and inaccuracies in the heuristics applied on them. Errors in the logs result from the presence of caches at various levels between a user's browser and the web server logs, which results in some user requests not making it to the web server. The format of the log files used also affects the accuracy of the heuristics. In this study for example, the log file was stored in the *common log* format, which is not as useful as the *extended common log* format, which has two additional fields: the *referer* and *user_agent* fields. The *referer* field indicates which URL a user had previously accessed before requesting the current page. This information can be used to more correctly attribute page accesses to the correct user. Likewise, the *user_agent* field, which refers to the generic name of browser from which a request is made, can be used to attribute the correct pages to two or more users with similar navigation behaviors, but with different browsers. In spite of the problems with the user access logs mentioned above, the association rules mining results showed several frequent navigation patterns, which suggested that these patterns were important.

Research Findings. Recommender system design involves the application of techniques from a variety of disciplines. This research made contributions in four areas: a new metric for web page classification; a new data mining algorithm; extension of previous work on hyperlink recommender that use past user navigation behaviors to profile current users by converting the problem of predicting user interests from a simple computation to a collaborative filtering problem with all its advantages; and an in-depth study of the behavior of CLARANS clustering algorithm in high dimensional space, and a comparison of CLARANS with PAM and CLARA, both of them related clustering algorithms. We summarize each of these contributions below.

Web Page Classification Study. Faced with the need of a metric that can be used to automatically classify web pages as content or navigation, where content pages are considered to be recommendable to a user, we sought to combine properties of individual web pages and the web site designer's view of important pages to determine this new metric. The LW ratio (ratio of number of outgoing links on a page to the word count of that page) was chosen as an appropriate property of a web page. The notion of information scent was considered appropriate in describing the importance the web site designer attaches to individual pages. Information scent is a subjective measure, but we reasoned that web pages that have relatively more incoming links than outgoing links are generally more important than pages with less, and so should have a larger scent. These ideas were captured using the PR metric. Hence PR and LW were combined to form the PR × ILW metric, which was computed for every page of the web site.

The small user study that was conducted suggested that PR alone does not help in classifying web pages; but PR × ILW does a better job than LW alone. A bigger user study would be required to authenticate this finding.

It should be noted that the LW and PR metrics are not new. What this research has contributed in the use of these metrics is the novel way in which they have been combined to generate a more powerful classification metric.

Data Mining. The contribution of this research in this area has been the development of a new implementation of the apriori algorithm that is influential in association rule mining. Our objective was to attempt to solve the major problem with the apriori algorithm: scalability. The classical implementation of the algorithm makes use of a join of k-1 frequent itemsets of the transactions database to itself, to obtain candidate k-itemsets, and the apriori property applied to the candidate itemsets to determine which of them are frequent.

The problem with the join is that if there are too many items in the database, then the results of the join may be too large to fit into main memory.

Our implementation of the apriori algorithm eliminates the join step by applying the apriori property during the *k*th iteration of the algorithm, to every record of the transactions database with itemset count greater than or equal to *k*. Because each transaction in the database almost always has fewer items than there are items in the database, the problem of a large join result in the classical algorithm is avoided.

Evaluation of the performance of the joinless apriori algorithm showed that it has better computation time for small values of minimum support count than the classical algorithm, and comparable computation time for large values of minimum support count.

Prediction Models for User Interests. We demonstrated in this research that association rules representing the correlations between user navigation behaviors and the pages they are interested in can be represented as an association rules matrix, with the navigation pages corresponding to user, the content pages corresponding to items rated by the users, and the rule confidence values corresponding to user ratings. The advantages of doing this include the following: prediction of the pages that a user with a given navigation behavior is interested in can be computed from the confidence scores of similar past navigation behaviors; the approach inherits the advantages inherent in collaborative filtering; the problems of collaborative filtering that stem from sparse ratings densities are avoided however, because the association rules matrix is already populated with the confidence values of most user navigation behaviors that are likely going to be encountered.

As was shown in Section 5.6, a prediction model that determined neighborhood vectors using the center-based approach, and that computed predictions of user interests using weighted average of neighborhood vectors, showed promising results. In this study, the similarity between vectors was computed using the cosine measure.

Clustering. Clustering is a frequently used technique in various stages of recommender system design. Although the CLARANS clustering algorithm was first proposed about a decade ago, there has not been much study on the behavior of the algorithm in high dimensional spaces of the type recommender systems operate in. Using the angle between document vectors as an appropriate measure for dissimilarity, it was demonstrated in this study that the quality of clusters discovered by CLARANS is better than those discovered by PAM and CLARA, although CLARA was fastest of the three algorithms.

Future Work. There are a number of directions that this work could proceed in. These include: determination of ranking of web pages that are judged to be interesting and recommendable, evaluation of generated recommendations, and the construction of a friendly user interface for the system. We consider each of these in turn.

Ranking of Recommendable Pages. The prediction scores of a user’s interest in web pages should not be the only factor that determines the pages’ relative position on the recommendation list. A page that is several mouse clicks away from the current page, for example, could be given a higher recommendation score than a page that is only one mouse click away, if the former has a lower prediction score. Link distance factor (Mobasher et al., 2000) for example, could be used as an appropriate weighting factor on the prediction score when recommendation scores are computed.

Evaluation of Recommended Pages. Ultimately, a recommender system’s acceptance would depend on users’ judgment of the recommendations they receive. As mentioned earlier, Prediction Model I that computed predictions of user interests using weighted average of neighborhood vectors, showed that the model correctly predicts user interests based on the available data. A user study would be required to determine if users find computed recommendations useful. Such a user study could be complicated by the fact that it may be influenced by the system interface.

One useful evaluation strategy may require the user to perform a task, for example, find all the web pages in the web site that provide information on a particular topic, say admissions. Meanwhile, in the background, the system computes recommendations based on the user’s changing navigation behavior and presents them in a window as he browses the site. The system could then keep track of the recommended hyperlinks that the user follows, and also presents a list of all the computed recommendations at the end of browsing session for the user to indicate which ones would have helped him navigate easily to the desired pages. If users find several of the system generated recommendations useful, that would be an indication that the system computes relevant recommendations.

User Interface Construction. After it is determined that the system generates relevant recommendations, user acceptance of the system may be enhanced with a good user interface. A good design should be simple and introduce as little distraction as possible from the screen the user would see if the recommendation component were absent.

APPENDIX A

Computation of Cosine Similarity Between Two Vectors

The cosine similarity measure is the cosine of the angle between two vectors, and is computed using the procedure outlined below.

Consider two vectors **A** and **B** in 2-dimensional space (Figure 53). The dot product between the vectors is defined as: $\mathbf{A.B} \quad AB\cos\theta.$

But $A_x \quad A\cos\alpha, B_x \quad B\cos\beta, \mathrm{A}_y \quad A\sin\alpha, B_y \quad B\sin\beta$ and $\theta = \alpha - \beta.$

It follows that:

$$\begin{aligned}\mathbf{A\cdot B} &= AB\cos(\alpha-\beta)\\ &= AB(\cos\alpha\cos\beta + \sin\alpha\sin\beta)\\ &= A\cos\alpha B\cos\beta + A\sin\alpha B\sin\beta\\ &= A_x B_x + A_y B_y\end{aligned}$$

In general, $\mathbf{X\cdot Y} = \sum_{i=1}^{n} x_i y_i$, where n is the dimension of the vectors. It follows from the definition of dot product that $\cos\theta \quad \frac{{}_{i\ 1}^{\ n} x_i y_i}{\sqrt{{}_{i\ \ 1}^{\ n}(x_i)^2}\sqrt{{}_{i\ \ 1}^{\ n}(y_i)^2}}$, since for any vector z, $z \quad \sqrt{{}_{i\ 1}^{\ n}(z_i)^2}$. Hence, given two vectors with vector representations **P** and **Q**, the cosine similarity measure between them is:

$$\cos\theta \quad \frac{{}_{i\ 1}^{\ n} w_{ip} w_{iq}}{\sqrt{{}_{i\ \ 1}^{\ n}(w_{iq})^2}\sqrt{{}_{i\ \ 1}^{\ n}(w_{ip})^2}} \quad \frac{{}_{i\ 1}^{\ n} w_{ip} w_{iq}}{\sqrt{{}_{i\ \ 1}^{\ n}(w_{iq})^2 \ {}_{i\ \ 1}^{\ n}(w_{ip})^2}},$$ where w_{ij} is the weight of term i in vector j.

If **P** and **Q** are binary vectors, the cosine similarity measure reduces to $\cos\theta \quad \frac{|P \quad Q|}{\sqrt{|P||Q|}}$.

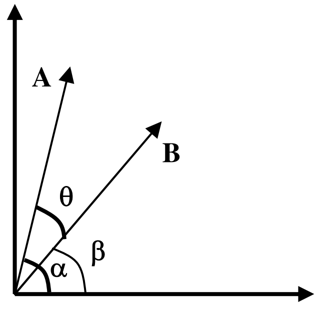


Figure 53. Illustrating computation of cosine similarity.

We notice that if the vector components represent the words in a document, the cosine measure normalizes for document lengths at the denominator. That helps in text processing, since long documents do not get an unfair advantage in the weighting of the terms in them. We next consider the steps involved in system design.

APPENDIX B

Stop List Used to Filter out HTML Page Presentation Code from Contents

&#198
&#193
&#194
&#192
&#197
&#195
&#196
&#199
&#208
&#201
&#202
&#200
&#203
&#205
&#206
&#204
&#207
&#209
&#211
&#212
&#210
&#216
&#213
&#214
&#222
&#218
&#219
&#217
&#220
&#221
&#225
&#226
&#230
&#224
&#229
&#227
&#228
&#231
&#233
&#234
&#232
&#240
&#235
&#237
&#238

&#236
&#239
&#241
&#243
&#244
&#242
&#248
&#245
&#246
&#223
&#254
&#250
&#251
&#249
&#252
&#253
&#255
&LT
&GT
&AMP
&QUOT
&NBSP
&REG
&COPY
&ENSP
&EMSP
&ENDASH
&EMDASH
<!--
<![
</A
</ABBR
</ABBREV
</ACRONYM
</ADDRESS
</APPLET
</AU
</AUTHOR
</B
</BANNER
</BDO
</BIG
</BLINK
</BLOCKQUOTE
</BODY

</BQ
</BUTTON
</CAPTION
</CENTER
</CITE
</CODE
</COL
</COLGROUP
</CREDIT
</DEL
</DEL
</DFN
</DIR
</DIV
</DL
</DT
</EM
</EMBED
</FIELDSET
</FIG
</FN
</FONT
</FORM
</FRAME
</FRAMESET
</H1
</H2
</H3
</H4
</H5
</H6
</HEAD
</HTML
</I
</IFRAME
</INS
</KBD
</LABEL
</LANG
</LEGEND
</LH
</LI
</LINK
</LISTING
</MAP

</MARQUEE
</MATH
</MENU
</NOBR
</NOFRAMES
</NOSCRIPT
</NOTE
</O:P
</OBJECT
</OL
</OPTGROUP
</OPTION
</P
</PERSON
</PLAINTEXT
</PRE
</Q
</S
</SAMP
</SCRIPT
</SELECT
</SMALL
</SPAN
</STRIKE
</STRONG
</STYLE
</SUB
</SUP
</TABLE
</TBODY
</TD
</TEXTAREA
</TITLE
</TR
</TT
</U
</UL
</VAR
</WBR
</XMP
<A
<ABBR
<ABBREV
<ACRONYM
<ADDRESS

<APPLET
<AREA
<AU
<AUTHOR
<B
<BANNER
<BASE
<BASEFONT
<BDO
<BGSOUND
<BIG
<BLINK
<BLOCKQUOTE
<BODY
<BQ
<BR
<BUTTON
<CAPTION
<CENTER
<CITE
<CODE
<COL
<COLGROUP
<CREDIT
<DD
<DEL
<DEL
<DFN
<DIR
<DIV
<DL
<DT
<EM
<EMBED
<FIELDSET
<FIG
<FN
<FONT
<FORM
<FRAME
<FRAMESET
<H1
<H2
<H3
<H4

<H5
<H6
<HEAD
<HR
<HTML
<I
<IFRAME
<IMG
<INPUT
<INS
<ISINDEX
<KBD
<LABEL
<LANG
<LEGEND
<LH
<LI
<LINK
<LISTING
<MAP
<MARQUEE
<MATH
<MENU
<META
<MULTICOL
<NOBR
<NOFRAMES
<NOSCRIPT
<NOTE
<O:P
<OBJECT
<OL
<OPTGROUP
<OPTION
<OVERLAY
<P
<PARAM
<PERSON
<PLAINTEXT
<PRE
<Q
<RANGE
<S
<SAMP
<SCRIPT
<SELECT
<SMALL
<SPACER
<SPAN
<SPOT
<STRIKE
<STRONG
<STYLE
<SUB
<SUB
<SUP
<SUP
<TAB
<TABLE
<TBODY
<TBODY
<TD
<TD
<TEXTAREA
<TEXTAREA
<TEXTFLOW
<TFOOT
<TH
<THEAD
<TITLE
<TR
<TT
<U
<UL
<VAR
<WBR
<XMP

BIBLIOGRAPHY

Aggarwal, C., & Srikant, R. (1994). *Fast algorithms for mining association rules.* Paper presented at the Proc. 20th Int. Conf. Very Large Data Bases, VLDB.

Agrawal, R., Imielinski, T., & Swami, A. (1993). *Mining association rules between sets of items in large databases.* Paper presented at the ACM SIGMOD International Conference on Management of Data.

Ahmad, M. A. W. (1999). Collecting user access patterns for building user profiles and collaborative filtering. In *4th International Conference on Intelligent User Interfaces* (pp. 57-64). Los Angeles, California: ACM Press, New York.

Aldenderfer, M. S., & Blashfield, R. K. (1984). *Cluster analysis* (Vol. 07-004). Newbury Park, US: SAGE.

Armstrong, R., Freitag, T., Joachims, T., & Mitchell, T. (1995). WebWatcher: A learning apprentice for the world wide web. In *Proceedings of AAAI Spring Symposium on Information Gathering from Heterogeneous Distributed Environments*.

Balabanovic, M., & Shoham, Y. (1997a). Content-based collaborative recommendation. *Communications of the ACM, 40,* 60-72.

Balabanovic, M., & Shoham, Y. (1997b). Fab: Content-based, collaborative recommendation. *Communications of the ACM, 40,* 66-72.

Baudisch, P., & Brueckner, L. (2002). TV Scout: Lowering the entry barrier to personalized TV program recommendation. In *In Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems (AH2002)*. Malaga, Spain.

Bauer, M., Pohl, W., & Webb, G. (1997). *Workshop on machine learning for user modeling*, from http://www.dfki.de/~bauer/um-ws/

Bauer, T., & Leake, D. (2002). Exploiting information access patterns for context-based retrieval. In *Proceedings of the 7th international conference on Intelligent user interfaces*. San Francisco, California, USA.

Belkin, N. (2000). Helping people find what they don't know. *Communications of the ACM, 43,* 58-61.

Belkin, N., & Croft, B. (1992, December 1992). Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM, 35,* 29-38.

Billsus, D., & Pazzani, M. (1999a). A hybrid user model for news story classification. In *Proceedings of the 7th International Conference on User Modeling*. Banff, Canada.

Billsus, D., & Pazzani, M. (1999b). A personal news agent that talks, learns and explains. In *Proceedings of the Third Annual Conference on Autonomous Agents* (pp. 268-275).

Billsus, D., & Pazzani, M. (2000). User modeling for adaptive news access. *User Modeling and User-Adapted Interaction, 10*(2-3), 147-180.

Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction, 6*(2-3), 87-129.

Brusilovsky, P. (2000). *Adaptive hypermedia: an attempt to analyse and generalize*, 2002, from http://wwwis.win.tue.nl/ah94/Brusilovsky.html

Brusilovsky, P. (2001). Adaptive hypermedia. *User Modeling and User-Adapted Interaction, 11*, 87-110.

Brusilovsky, P., & Maybury, M. (2002). From adaptive hypermedia to the adaptive web. *Communications of the ACM, 45*(5), 30-33.

Burke, R. (2001). Knowledge-based recommender systems. In *Encyclopedia of library and information science* (Vol. 69/supl. 32).

Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction, 12*(4), 331-370.

Card, S. K., Pirolli, P., Van Der Wege, M., Morrison, J. B., Reeder, R. W., Schraedley, P. K., et al. (2001). Information scent as a driver of Web behavior graphs: Results of a protocol analysis method for Web usability. In *Proceedings of the Conference on Human factors in computing systems, CHI '01* (pp. 498-505). Seattle, WA: ACM Press.

Catledge, L., & Pitkow, J. (1995). Characterizing browsing strategies in the world wide web. *Computer Networks and ISDN Systems, 27*(6), 1065-1073.

Chen, M. S., Park, J. S., & Yu, P. S. (1996). Data mining for path traversal patterns in a Web environment. In *Proceedings of the 16th International Conference on Distributed Computing Systems* (pp. 385-392).

Cingil, I., Dogac, A., & Azgin, A. (2000, August 2000). A broader approach to personalization. *Communications of the ACM, 43,* 136-142.

Claypool, M., Brown, D., Le, P., & Waseda, M. (2001). Inferring user interest. *IEEE Internet Computing, 5,* 32-39.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., & Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*. University of California, Berkeley , USA.

Claypool, M., Le, P., Waseda, M., & Brown, D. (2001). Implicit interest indicators. In *6th international conference on Intelligent user interfaces* (pp. 33-40). Santa Fe, New Mexico, United States.

Clemen, R. T. (1996). *Making hard decisions: An introduction to decision analysis* (2nd ed.). Belmont CA, USA: Duxbury Press.

Cooley, R., Mobasher, B., & Srivastava, J. (1997). Web mining: Information and pattern discovery on the World Wide Web. In *International Conference on Tools With Artificial Intelligence* (pp. 558-567). Newport Beach, CA, USA.

Cooley, R., Mobasher, B., & Srivastava, J. (1999). Data preparation for mining world wide web browsing patterns. *Journal of Knowledge and Information Systems, 1*(1), 1999.

Craven, P. (2003). *Google's PageRank explained and how to make the most of it*. Retrieved 1 July, 2003, from http://www.webworkshop.net/pagerank.html

Deerwester, S., Dumais, T. S., Furnas, G., Landauer, T., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science, 41*(6), 391-407.

Farber, D. (2001a). Balancing security and liberty. *IEEE Internet Computing, 5,* 96-96.

Farber, D. (2001b). *Balancing security and liberty*, 2001, from http://computer.org/internet/v5n6

Fielding, R., Gettys, J., Mogul, J., Frystk, H., Masinter, L., Leach, P., et al. (Artist). (1999). *Hypertext transfer protocol -- HTTP/1.1*

Folz, W. P., & Dumais, T. S. (1992, December 1992). Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM, 35,* 51-218.

Furnas, G. (1997). Effective view navigation. In *Proceedings of the Human Factors in Computing Systems, CHI '97* (pp. 367-374). Atlanta, GA: ACM Press.

GroupLens. (2002). *Movielens*, 2002, from http://www.movielens.umn.edu/login

Guha, S., Rastogi, R., & Shim, K. (1998). CURE: An efficient clustering algorithm for large databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data* (pp. 73--84). Seattle, WA.

Han, J., & Kamber, M. (2001). *Data mining: Concepts and techniques*. San Diego, CA, USA: Academic Press.

Hanani, U., Shapira, B., & Shoval, P. (2001). Information filtering: Overview of issues, research and systems. *User Modeling and User-Adapted Interaction, 11*, 203-259.

Hill, C. W., Hollan, D. J., Dave, W., & McCandless, T. (1992). Edit wear and read wear. In *Conference on Human Factors and Computing Systems*. Monterey, California, United States.

Hirashima, T., Hachiya, K., Kashihara, A., & Toyoda, J. (1997). Information filterng using user's context on browsing in hypertext. *User Modeling and User-Adapted Interaction, 7*, 239-256.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., & Rommelse, K. (1998, July 1998). *The lumiere project: Bayesian user modeling for inferring the goals and needs of software users.* Paper presented at the Fourteenth Conference on Uncertainty in Artificial Intelligence.

James, J. (2003). Bayes' theorem. In E. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Vol. forthcoming).

Karypis, G., Han, E.-H., & Kumar, V. (1999). Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer: Special Issue on Data Analysis and Mining, 32*(8).

Kaufman, L., & Rousseeuw, P. J. (1990). Finding groups in data: An introduction to cluster analysis. New York: John Wiley & Sons.

Kelly, D., & Belkin, N. (2001). Reading time, scrolling and interaction. In *24th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 408-409).

Kim, J., Oard, W. D., & Romanik, K. (2001). User modeling for information filtering based on implicit feedback. In *ISKO-France 2001*. Nanterre, France.

Konstan, A. J., Miller, N. B., Maltz, D., Herlocker, L. J., Lee, R. G., & Riedl, J. (1997, March 1997). GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM, 40,* 77-87.

Krulwich, B. (1997). Lifestyle finder: Intelligent user profiling using large-scale demographic data. *AI Magazine, 18,* 37-45.

Lang, K. (1995). Newsweeder: Learning to filter news. In *12th International Conference on Machine Learning* (pp. 331-339). Lake Tahoe, CA, USA.

Larson, K., & Czerwinski, M. (1998). *Web page design: Implications of memory, structure, and scent for information retrieval.* Paper presented at the CHI'98 Human Factors in Computing Systems.

Lieberman, H. (1995). Letzia: An agent that assists Web browsing. In *In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence.* (pp. 924--929.). Montreal, Canada.

Lieberman, H., Fry, C., & Weitzman, L. (2001). Exploring reconaissance agents. *Communications of the ACM, 44*(8).

Lyman, P., & Varian, H. R. (2004). *How much information 2003?* Retrieved December 31, 2004

Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM, 37.*

Malone, T., Grant, K., Turbak, F., Brobst, S., & Cohen, M. (1987). Intelligent information sharing systems. *Communications of the ACM, 30,* 390-402.

Mannila, H., Toivonen, H., & Verkamo, I. (1994). Efficient algorithms for discovering association rules. In *AAAI Workshop on Knowledge Discovery in Databases (KDD-94).*

Marchionini, G. (1995). *Information seeking in electronic environments*. Cambridge, New York: Cambridge University Press.

McCarthy, J. F., & Anagnost, T. D. (1998). MUSICFX: An arbiter of group preferences. In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work* (pp. 363-372). Seattle, WA.

McKenna, E., & Smyth, B. (2000). Competence-guided editing methods for lazy learning. In *Proceedings of the 14th European Conference on Artificial Intelligence*. Berlin.

Middleton, E. S., Shadbolt, N., & De Roure, C. D. (May 2002). Exploiting synergy between ontologies and recommender systems, *The Eleventh International World Wide Web Conference (WWW2002).* Hawaii, USA.

Middleton, S., De Roure, D., & Shadbolt, N. (2001). Capturing knowledge of user preferences: ontologies in recommender systems. In *International Conference On Knowledge Capture* (pp. 100-107). Victoria, British Columbia, Canada: ACM Press, New York.

Miyahara, K., & Pazzani, M. (2000). *Collaborative filtering with the simple bayesian classifier.* Paper presented at the The Sixth Pacific Rim International Conference on Artificial Intelligence (PRICAI), Melbourne, Australia.

Mobasher, B., Cooley, R., & Srivastava, J. (2000, August 2000). Automatic personalization based on Web usage mining. *Communications of the ACM, 43,* 142-151.

Mobasher, B., Dai, D., Luo, L., & Nakagawa, M. (2001). Effective personalization based on association rule discovery from web usage data. In *Third international workshop on Web information and data management* (pp. 9-15).

Morita, M., & Shinoda, Y. (1994). Information filtering based on user behavior analysis and best match text retrieval. In *Seventeenth annual international ACM-SIGIR conference on Research and development in information retrieval* (pp. 272-281).

Mulvenna, D. M., Anand, S. S., & Büchner, G. A. (2000, August 2000). Personalization on the net using web mining. *Communications of the ACM, 43,* 123-125.

Nentwich, M. (2002). *Lost in hyperspace*, from http://www.oeaw.ac.at/ita/ebene5/dsk/APSA/ lost_in_hyperspace.htm

Ng, R., & Han, J. (1994). Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases* (pp. 144--155). Santiago, Chile: Morgan Kaufmann Publishers.

Ng, R., & Han, J. (2002). CLARANS: A method for clustering objects for spatial data mining. In *IEEE Transactions on Knowledge and Data Engineering* (Vol. 14, pp. 1003-1016).

Nichols, D. (1998). Implicit rating and filtering. In *Fifth DELOS Workshop on Filtering and Collaborative Filtering* (pp. 31-36). Budapest, Hungary.

Nielsen, J. (1995). *Multimedia and hypertext: The internet and beyond*: Academic Press Professional.

Nkweteyim, D. L. (2005). Web page classification based on web page size and hyperlinks and website hyperlink structure. In H. Gamboa & A. Fred (Eds.), *Fifth International Workshop on Pattern Recognition in Information Systems – PRIS 2005, 7th International Conference on Enterprise Information Systems*. Miami, FL.

Nkweteyim, D. L., & Hirtle, S. C. (2005). A new joinless apriori algorithm for mining association rules. In H. Gamboa & A. Fred (Eds.), *Fifth International Workshop on Pattern Recognition in Information Systems – PRIS 2005, 7th International Conference on Enterprise Information Systems*. Miami, FL.

Oard, W. D. (1997). The state of the art in text filtering. *User Modeling and User-Adapted Interaction, 7*, 141-178.

Oard, W. D., & Kim, J. (1998). Implicit feedback for recommender systems. In *AAAI Workshop on Recommender Systems*. Madison, WI, USA.

O'Connor, M., Cosley, D., Konstan, A. J., & Riedl, J. (2001). PolyLens: A recommender system for groups of users. In *In Proceedings of ECSCW 2001* (pp. 199-218). Bonn, Germany.

Pazzani, M. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review, 13*(5/6), 393-408.

Pazzani, M., & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting Web sites. *User Modeling and User-Adapted Interaction, 27*, 313-331.

Pazzani, M., Muramatsu, J., & Billsus, D. (1996). Syskill & Webert: Identifying interesting Web sites. In *Proc. 13th Nat'l Conf. AI AAAI 96* (pp. 54--61). Menlo Park, California.: AAAI Press.

Perkowitz, M., & Etzioni, O. (2000, August 2000). Adaptive Web sites. *Communications of the ACM, 43,* 152-158.

Pirolli, P. (1997). *Computational models of information scent-following in a very large browsable text collection.* Paper presented at the CHI ’97 Human Factors in Computing Systems, Atlanta, GA.

Pirolli, P., & Fu, W. (2003). SNIF-ACT: A model of information foraging on the World Wide Web. In P. Brusilovsky, A. Corbert & F. De Rosis (Eds.), *User Modeling 2003*.

Pirolli, P., Pitkow, J., & Rao, R. (1996). Silk from a sow's ear: Extracting usable structures from the Web. In *Conference on Human Factors in Computing Systems (CHI-96)*.

Pitkow, J. (1997). In search of reliable usage data on the WWW. In *Proceedings of the Sixth International WWW Conference* (pp. 1343-1355).

Porter, M. (1980). An algorithm for suffix stripping. *Program, 14*, 130-137.

Rafter, R., Bradley, K., & Smyth, B. (2000). Automated collaborative filtering applications for online recruitment services. In *AH2000* (pp. 363-368). Trento, Italy: Springer.

Ramakrishnan, N., Keller, B., Mirza, B., Grama, A., & Karypis, G. (2001, November-December 2001). Privacy risks in recommender systems. *IEEE Internet Computing,* 54-62.

Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM, 40,* 56-58.

Rich, E. (1979). User modeling via stereotypes. *Cognitive Science, 3*, 329-354.

Riedl, J. (2001, November/December 2001). Personalization and privacy. *IEEE Internet Computing, 5,* 29-31.

Rocchio, J. (1971). Relevance feedback in information retrieval. In G. Salton (Ed.), *The Smart retrieval system: Experiments in automatic document processing* (pp. 313-323). Englewood, NJ: Prentice-Hall.

Rogers, I. (2003). *The Google pagerank algorithm and how it works*. Retrieved 1 July, 2003, from http://www.iprcom.com/papers/pagerank/

Rucker, J., & Polanco, M. (1997). SiteSeer: Personalized navigation for the Web. *Communications of the ACM, 40,* 73-75.

Rumelhart, D., Hinton, R., & Williams, R. (1986). Learning internal representations by error propagation. In D. Rumelhart (Ed.), *Parallel Distributed Processing I* (pp. 318-362): MIT Press.

Salton, G. (1989). *Automatic text processing*. Reading, Massachusetts: Addison-Wesley.

Salton, G., & McGill, W. J. (1983). *Introduction to modern information retrieval*. New York: McGraw-Hill.

Salton, G., & Wong, A. (1975). A vector space model for automatic indexing. *Communications of the ACM, 18*.

Sander, J., Ester, M., Kriegel, H., & Xu, X. (1998). Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery, 2*(2).

Sarwar, B., Karypis, G., Konstan, A. J., & Riedl, J. (2000a). Analysis of recommendation algorithms for e-commerce. In *ACM Conference on Electronic Commerce* (pp. 158-167).

Sarwar, B., Karypis, G., Konstan, A. J., & Riedl, J. (2000b). Application of dimensionality reduction in recommender systems. In *Web Mining for E-Commerce*. Boston, MA, USA.

Sarwar, B., Karypis, G., Konstan, A. J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the tenth international conference on World Wide Web* (pp. 285-295). Hong Kong.

Sarwar, B., Konstan, A. J., Borchers, A., Herlocker, L. J., Miller, N. B., & Riedl, J. (1998). Using filtering agents to improve prediction quality in the GroupLens research collaborative filtering system. In *In Proc. ACM Conf. Computer Support Cooperative Work (CSCVO 1998)*.

Schafer, J. B., Konstan, A. J., & Riedl, J. (2001). E-commerce recommendation applications. *Data Mining and Knowledge Discovery, 5*(1), 115-153.

Schafer, J. B., Konstan, J., & Riedl, J. (1999). Recommender systems in e-commerce. In *First ACM conference on Electronic commerce* (pp. 158-166). Denver, Colorado, USA: ACM Press, New York, USA.

Schechter, S., Krishnan, M., & Smith, M. (1998). Using path profiles to predict HTTP requests. In *7th International World Wide Web Conference*. Brisbane, Australia.

Shani, G., Brafman, R., & Heckerman, D. (2002). An MDP-based recommender system. In *Proceedings of Eighteenth Conference on Uncertainty in Artificial Intelligence*. Edmonton, Alberta, Canada: Morgan Kaufmann Publishers.

Shenk, D. (1997). *Data smog: Surviving the information glut*. New York: Harper Collins.

Singh, M. P. (2001). Privacy and Open Services. *IEEE Internet Computing,* 4-5.

Singh, M. P. (2002). Privacy for telecom services. *IEEE Internet Computing, 6,* 4-5.

Smyth, B., & Cotter, P. (2000, August 2000). A personalized television listings service. *Communications of the ACM, 43,* 107-111.

Swingler. (1996). *Applying neural networks: A practical guide*. San Diego, California: Academic Press.

Thimbleby, H., Jones, M., & Theng, Y. L. (1997). Is "lost in hyperspace" lost in controversy? In *Hypertext'97*. Southampton, UK.

Towle, B., & Quin, C. (2002). *Knowledge based recommender systems using explicit user models*, from http://www.igec.umbc.edu/kbem/towle.pdf

Ullman, J. D. (2003a). *Clustering*. Retrieved December 27, 2004, from http://www-db.stanford.edu/~ullman/mining/cluster1.pdf

Ullman, J. D. (2003b). *Clustering, part I*. Retrieved February 10, 2004, from http://www-db.stanford.edu/~ullman/cs345notes/cs345-cl.pdf

Ullman, J. D. (2003c). *Clustering, part II*. Retrieved December 27, 2004, from http://www-db.stanford.edu/~ullman/cs345notes/cs345-cl2.pdf

Ungar, L., & Foster, D. (1998). Clustering methods for collaborative filtering. In *AAAI Workshop on Recommendation Systems*.

Varian, H. R. (1995). The information economy: How much will two bits be worth in the digital marketplace? *Scientific American,* 200-201.

W3C. (2003). *The common logfile format*, from http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format

Webb, G., Pazzani, M., & Billsus, D. (2001). Machine learning for user modeling. *User Modeling and User-Adapted Interaction, 11*, 19-29.

Wei, W., Yang, J., & Muntz, R. (1997). STING: A statistical infromation grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Databases* (pp. 186--195).

Wikipedia. (2004). *Information overload*. Retrieved December 31, 2004, from http://en.wikipedia.org/w/index.php?title=Information_overload&action=history

Yan, T., Jacobsen, M., Garcia-Molina, H., & Dayal, U. (1996). From user access patterns to dynamic hypertext linking. In *5th World Wide Web Conference*. Paris.

Yongling, D. (2004). *Pure statistical analysis of security returns*, from http://mywebpages.comcast.net/ylding/returns/html/node4.html#SECTION00041000000000000000

Zhang, T., Ramakrishnan, R., & Livny, M. (1996). BIRCH: An efficient data clustering method for very large databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data* (Vol. 34, pp. 103--114). Montreal.

Zimdars, A., Chickering, D. M., & Meek, C. (2001). Using temporal data for making recommendations. In *Proceedings of Seventeenth Conference on Uncertainty in Artificial Intelligence* (pp. 580-588). Seattle, WA: Morgan Kaufmann Publishers.

Zukerman, I., & Albrecht, W. (2001). Predictive statistical models for user modeling. *User Modeling and User-Adapted Interaction, 11*(1-2), 5-18.