A NEW CLASS OF TECHNIQUES FOR WEB PERSONALIZATION

BHUSHAN SHANKAR SURYAVANSHI

A THESIS

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE AND SOFTWARE ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

MARCH 2006

Your file Votre référence
ISBN: 0-494-14338-X
Our file Notre référence
ISBN: 0-494-14338-X

ABSTRACT

A NEW CLASS OF TECHNIQUES FOR WEB PERSONALIZATION

BHUSHAN SHANKAR SURYAVANSHI

Web personalization aims to provide content and services tailor-made to the needs of individual users usually from the knowledge gained through their (previous) interactions with the site. Typically, an access behavior model of users is learnt from the usage of the website which is then used to provide personalized recommendations to the current user(s). Clearly the performance of a recommender system will depend on the underlying model. Two fundamental challenges in personalization are effectiveness (accuracy) and efficiency (scalability) of recommender algorithms. In this thesis, we present a new class of techniques for efficient and effective web personalization. All the techniques are based on a new algorithm for fast mining of web usage data, called Relational Fuzzy Subtractive Clustering (RFSC). RFSC is scalable to large datasets, does not require user biased control parameters, is relatively more immune to noise which is inherent in usage data, and can capture overlapping user interest areas. Making innovative use of fuzzy grade of memberships and fuzzy prototypes of the access behavior model learnt through RFSC, we also propose improvements over cluster based and association rule based recommender models. Since browsing behavior of users on the web is not static but changes dynamically over time, we further propose a new maintenance scheme, which extends RFSC, to efficiently add new web usage data to an existing model in order to achieve adaptability over non-stationary volatile web environment. We validate our claims of effectiveness and efficiency through extensive experimentation on synthetic data as well as large datasets of web logs.

Acknowledgments

I express my sincere gratitude and appreciation to my supervisors Dr. Sudhir P. Mudur and Dr. Nematallaah Shiri. Thanks to Dr. Mudur for an endless supply of insight, inspiration and for suggesting several of the possible counter-arguments to my own that are considered in this thesis. Thanks to Dr. Shiri, for having unwavering faith in me, and whose advice, wit, and keen ability of observation I will remember long after I graduate. It is through numerous interactions and meetings with my supervisors that I have gained a solid understanding of how to and how not to research an area. I further extend a special thanks to them for giving me opportunities to present papers, which we co-authored, at international conference/ workshops, which was an experience in itself.

I also wish to thank my colleagues in the database lab for sharing ideas, knowledge and discussions together. These include Ahmed Alasoud, Mihail Halatchev, Ali Kiani, Aida Nemalhabib, Ali Taghizadeh, and Anand Thamildurai. I thank my friends, Pranav Ambole, Pranati Asgaonkar, Anirban Mazumdar, Sarang Patil, Akshay Kumar, Anil Kumar V., and Shilpa Shrimal, for being a constant source of support and encouragement.

I thank my parents and my sister for their unconditional love, continuous support and for all their sacrifices; without which I wouldn't be here.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The World Wide Web is an ever growing, most frequently visited market place for e-services and products. The vast variety of products and services on the Internet has both simplified and complicated shopping online. While ordering has been rendered more convenient, selection has become more difficult for users by the sheer increase in choices. Users are often overwhelmed with the amount of information, as it is much more than what can be easily interpreted and absorbed. Another major problem is disorientation, as users generally feel lost in this huge information space. Personalized recommendation can alleviate these problems to a large extent. It can be compared to having your favorite bookseller pull out a copy of a book he just knows you would really like. Web personalization aims to provide content and services tailor-made to the needs of individual users usually from the knowledge gained through their (previous) interactions with the site, just in the same way a sales person understands the needs of a customer by interacting with the customer. The goal is to improve the user's experience of an e-service. Personalization is a "you-and-me" relationship-building activity, which includes recognition, understanding, and serving the specific needs of users and customers. As described by Jeff Bezos, CEO of Amazon.com, "If I have 3 million customers on the web, I should have 3 million stores on the web" [Schafer et al., 1999].

A distinction is often made between customization and personalization. Intuitively, the former is a user-driven process while the latter is system-driven. In customization, the user can configure the layout and structure of the site. The control of the look and/or content is explicit, and the user is in control and is actively involved in the process. In personalization, on the other hand, the user is seen as passive, or at least somewhat less in control. It is the website that monitors, analyses, and reacts to user's behavior dynamically.

The web personalization process can be divided into four phases [Mobasher et al., 2000; Nasraoui et al., 2005]. The first three phases are collecting, preprocessing, and analysis of web data, and the final phase is recommendation. While the first two phases essentially gather and prepare the web data for further processing, the analysis phase attempts to obtain (or understand) a model of user behavior, user group characteristics, and user preferences. The results of this analysis are then used by a recommendation engine to provide personalized recommendations to the user, say, adding hyperlinks depending on user preference models to the last web page requested by the user. This process is discussed in detail in Chapter 2.

The web personalization process relies on one or more of the following types of data [Srivastava et al. 2000]:

- Content: The real data in the web pages. This can be simple text, images, or structured data, such as information retrieved from databases.
- Structure: It represents the organization of the content. The content can be either data entities used within a web page, such as HTML or XML tags, or data entities

used to put a web site together, such as hyperlinks connecting one page to another.

- Usage data: This represents a web site's usage, such as visitor's IP address, time and date of access, complete path (files or directories) accessed, referrers' address, and other attributes that can be included in a Web access log.

In our work, we make use of the web usage data of the site and learn usage profiles from such data. It is these usage profiles that will be made use of in a recommender system.

1.1 Data Mining on the Web

Data mining is a nontrivial process of discovering valid, previously unknown, potentially useful and ultimately understandable patterns in data. Web mining is the use of data mining techniques to automatically discover and extract information from web documents and services. It can be further divided into content mining, structure mining, and usage mining, depending on the type of data being mined. Web usage mining is the application of data mining techniques to discover usage patterns from web data, in order to understand and better serve the needs of web-based applications. Below we discuss different techniques that can be applied to web usage data [Srivastava 2000].

1) Statistical Analysis

Statistical techniques are the most common methods used to extract knowledge about visitors to a web site. One can perform different kinds of descriptive statistical analysis (frequency, mean, median, etc.) on variables such as page views, viewing time and length of a navigational path. Many web traffic analysis tools produce a periodic report containing statistical information such as most frequently accessed pages, average view time of a page or average length of a path through a site, etc. This report may include

limited low-level error analysis such as detecting unauthorized entry points or finding most common invalid URL. Despite lacking in the depth of analysis, this type of knowledge can be potentially useful for improving system performance, enhancing security of the system, facilitating site modification task, and providing support for marketing decisions.

2) Association Rules

Association rule generation can be used to relate pages in a website that are most often referenced together in a single server session. In the context of web usage mining, association rules refer to sets of pages that are frequently accessed together. These pages may not be directly connected to one another via hyperlinks. For example, association rule discovery may reveal a correlation between users who visited a page containing electronic products to those who access a page about sporting equipment. Aside from applicability in business and marketing applications, the presence or absence of such rules can help web designers to restructure their web site. Association rules may also serve as heuristics for prefetching documents in order to reduce user-perceived latency when loading a page from a remote site.

3) Clustering

Clustering is a technique to group together a set of items having similar characteristics. In the context of web usage domain, there are two kinds of interesting clusters to be discovered: usage clusters and page clusters. Clustering of users tends to identify groups of users exhibiting similar browsing patterns. Such knowledge is especially useful for inferring user demographics in order to perform market segmentation in E-commerce applications or to provide personalized web content to the users. On the other hand,

clustering of pages will discover groups of pages with somehow related content. This information is useful for Internet search engines and web assistance providers. In both applications, permanent or dynamic HTML pages can be created that suggest related hyperlinks to the user according to the user's query or past history of information needs.

4) Classification

Classification is the task of mapping a data item into one of several predefined classes. In the web domain, one is interested in developing a profile of users belonging to a particular class or category. This requires extraction and selection of features that best describe the properties of a given class or category. For example, classification on server logs may lead to the discovery of interesting rules such as: 30% of users who placed an online order in /Product/Music are in the age group 18-25 and live on the west coast.

5) Sequential Patterns

Sequential pattern discovery attempts to find patterns such that presence of a set of items is followed by another item in a time-ordered set of episodes (pages). This approach can be used by web marketers to predict future visits, which can then be helpful in placing advertisements aimed at certain user groups. Other types of temporal analysis that can be performed on sequential patterns include trend analysis, change point detection, and similarity analysis.

1.2 Recommender Systems

Social information filtering exploits similarities between the tastes of different users to recommend (or advise against) items. It relies on the fact that people's tastes are not randomly distributed: there are general trends and patterns within the taste of a person and as well as amongst groups of people. Social information filtering automates the

process of "word-of-mouth" recommendations. A significant difference is that, instead of having to ask a number of friends about a few items, a social information filtering system can consider thousands of other people, and consider thousands of different items, all happening autonomously and automatically. [Shardanand and Maes, 1995].

Examples of such applications include recommending books, CDs, and other products at Amazon.com [Linden et al., 2003], movies by MovieLens [Konstan et al., 1997], news at VERSIFI Technologies [Billsus et al., 2002], and music by Ringo [Shardanand and Maes, 1995].

The recommendation problem is to estimate ratings for the items that have not been seen by a user [Adomavicius and Tuzhilin, 2005]. Intuitively, this estimation is usually based on the ratings given by this user to other items and on some other information that we shall describe later. Once we can estimate ratings for the yet unrated items, we can recommend to the user, items with the highest estimated ratings. There are three basic approaches used in recommender systems: content-based filtering, rule-based filtering, and collaborative filtering [Eirinaki and Vazirgiannis 2003, Nasraoui et al., 2005], described briefly as follows.

- Content-based filtering is solely based on individual users' preferences. It tracks each user's behavior and recommends items that are similar to the ones the user preferred in the past. Such systems exploit the product information, say, attributes of specific domain items, e.g., author and subject for books, and artist and genre for music items. They do not require any previous implicit or explicit user rating or purchase data to make recommendations. The user will be recommended items similar to the ones the user preferred in the past. Content-based techniques are

limited by the features that are explicitly associated with the objects that they recommend. Hence in order to have a sufficient set of features, the content must either be in a form that can be parsed automatically (e.g., text) or the features should be assigned manually [Adomavicius and Tuzhilin, 2005].

- In rule-based filtering, the user has to answer some questions, until s/he receives a customized result. It requires heavy planning and customization by expert, lacks intelligence, and tends to be static. It also requires a careful selection of the questionnaire.
- Collaborative filtering (CF) is the most successful and widely used technique in building recommender systems [Sarwar et al., 2000b]. The goal of CF is to predict the preferences of a user, referred to as *active user*, based on the preference of a group of "like-minded" users. The key idea is that the active user will prefer those items that "like-minded" people prefer or even the ones that dissimilar people do not prefer. This approach relies on history, a collection of all previous users' interests, which could be inferred from user's ratings of the items at a website (products or web pages). Basically the vast amount of historical information accumulated is queried based on the current navigational pattern of the active user to provide him/her a personalized experience. Various CF techniques are explained in detail in Chapter 2.

The above filtering techniques may also be combined in different ways to overcome the limitations of individual techniques [Burke, 2002]. In this thesis, we focus on CF techniques to support personalized recommendations.

1.3 Motivation

[Breese et al., 1998] identified two major classes of CF algorithms: memory-based and model-based. Memory-based algorithms [Breese et al. 1998, Konstan et al., 1997, Shardanand and Maes, 1995, Delgado and Ishii, 1999] process the entire recorded dataset of user preferences to make predictions. These algorithms employ a notion of distance to find a set of users, known as neighbors, which tend to agree with the active user. The preferences of neighbors are then combined to produce a prediction or top-N recommendation for the active user. The preferences could be combined in a linear fashion or by weighting. Model-based methods on the other hand [Breese et al. 1998, Hofmann 2003, Ungar and Foster,1998, Sarwar et al., 2000b] use the recorded dataset to estimate or learn a model offline, which is then used for predictions. Popular web usage mining techniques such as clustering, association rule mining, and sequence pattern discovery have been used for this purpose [Mobasher, 2004, Nasraoui et al., 2005]. These techniques extract characteristics (patterns) or usage profiles from the usage history data through an offline process and employ these patterns to generate recommendations in an online process. Usage profiles capture different interests and trends among users accessing the site. These usage profiles can be used for recommendations, path prediction, perfecting of pages, better structuring of web sites, and in a nutshell, improving the experience of the user browsing the website for information and/or services. Usage profile information can also be used by content creators to understand which material is used more, how long the material is viewed by which types of users, and in what order they are accessed, etc. The set of profiles reflect the semantics of the user history database at that point of time.

Two important challenges in CF-based recommender systems are *accuracy* and *scalability*. Memory-based techniques are simple to understand, implement, and use. While they provide high accuracy recommendations and admit easy addition of new data, they are computationally expensive as the size of the input dataset increases. These techniques can be used to search typically up to tens of thousand of potential neighbors in real-time. Modern e-commerce applications, on the other hand, require searching tens of millions of potential neighbors [Sarwar, 2000b]. Due to the large online search cost involved, memory-based CF techniques do not scale up to handle such large data volumes. However, model-based techniques reduce the online processing cost by making use of the usage profiles learnt in the offline-modeling process. The improvement in online processing performance often comes at the cost of reduced accuracy of recommendations as model-based CF makes use of a general model for the entire preference dataset rather than the actual data, as is the case in memory-based CF.

The performance of a recommender system will depend on the underlying model, if one is used. Each popular website is visited by a large number of users with variety of needs and goals. A critical factor in the choice of a modeling technique is its scalability to large datasets. For most of the earlier proposed modeling techniques, the time required to compile the data into a model can be prohibitive for large datasets, thus making model-based CF techniques also less practical [Pennock et al., 2000]. Moreover, a desired modeling technique should be immune to noise, which is inherently present in web usage data. The browsing behavior of users on the web is highly uncertain and fuzzy in nature. A user may browse the same page for different purposes. Each time the user accesses the site, he or she may have different browsing goals. Furthermore, the same user in the same

session may have different goals and interests at different times. A modeling technique should be able to capture such overlapping user interests. It should also be independent of user specified control parameters, which may bias the modeling process. All the aforementioned factors make unsupervised classification or clustering more desirable for learning the access behavior model. Therefore, in our work we focus on clustering of web usage data for discovery of usage profiles which adequately model user access behavior on that web site. For comparison purposes, we shall also consider association rule based models.

1.4 Contributions

The mining community has focused considerably on developing efficient algorithms for finding patterns. However, it is also very important to research ways in which these patterns can be applied and used. In addition to our focus on mining knowledge from web usage data efficiently, we will also consider applying this knowledge for giving the user a personalized experience.

The contributions of this research are as follows.

1. **Relational Fuzzy Subtractive Clustering**. We first propose a technique based on fuzzy clustering for extracting usage profiles from huge web access logs so as to capture the fuzziness and uncertainty in user behavior into soft classes. We call it as Relational Fuzzy Subtractive Clustering (RFSC) [Suryavanshi *et al.* 2005a]. Our technique has the following advantages over traditional fuzzy clustering algorithms.
    - *Scalability*: It scales to large datasets and reduces the concern of prohibitively large model compilation time.

- *Parameter Independence*: Does not require any user biased control parameters. This is clearly an advantage since otherwise the clustering may not manifest the true structure in the data.
- *Immunity to noise*: RFSC is highly immune to noise, which is inherent in web data.
- *β Transforms*: Does not require expensive β spread transformations for converting the non-Euclidean Relational data into Euclidean space [Hathway and Bezdek, 1994].

We also propose a validity index for RFSC, called *index of goodness,* to measure the quality of clustering based on the popular Xie-Beni index [Xie and Beni, 1991]. We have conducted a comprehensive comparative study of RFSC and Fuzzy C-means [Bezdek, 1982] based relational clustering. For this, we have considered various factors including need of user specified control parameters for clustering, scalability to larger datasets, immunity to noise which is the inherent nature of web clickstreams and the ability to incorporate new additional data points into an existing cluster model. We also measure cluster similarity between the partitions obtained by the two techniques using the popular Rand coefficient as well as by comparing their usage profiles. To the best of our knowledge, this is the first such comprehensive experimental study, providing us with valuable measures to objectively judge the performance of these techniques. In addition to throwing up a number of interesting observations, our study also provides a basis for future research and applications of the proposed concepts and techniques.

2. **Fuzzy Hybrid Collaborative Filtering**. As discussed earlier, two fundamental challenges to a CF-based recommender system are accuracy and scalability. While memory-based methods have a high accuracy, they become prohibitively expensive to use as the size of the input dataset increases. The model-based approach has a complementary advantage: while it reduces the online processing cost, it often comes at the cost of reduced accuracy of recommendations. An important contribution of this thesis is a Fuzzy Hybrid CF technique [Suryavanshi et al. 2005b] that approaches the accuracy of memory-based and the scalability of model-based CF, and hence inheriting the advantages of both. This is achieved by utilizing in an innovative way, properties of the underlying modeling technique, namely Relational Fuzzy Subtractive Clustering. We make use of the fuzzy prototypes and the fuzzy grade of memberships obtained from the RFSC model. The fuzzy nearest prototype of the active user is used to find a group of like-minded users within which a memory-based search is conducted. We also extend our technique to fuzzy K-nearest prototypes (K-NP). Moreover, we also show how equal depth binning can further increase the scalability of our approach.
3. **Two Level Model-Based Collaborative Filtering.** Association rule based recommender systems often suffer in accuracy due to the sparse nature of web usage data. Top-level navigational pages (visited more frequently) may undermine the lower level content pages in the site hierarchy. Using RFSC as the first level modeling and then mining association rules within individual clusters, we develop a two level model-based CF technique [Suryavanshi et al., 2005c]. We show how segregation of interest areas could help in giving more relevant and

pertinent recommendations to the current user. Our technique achieves better accuracy than pure association rule based recommendation model.

4. **Incremental Relational Fuzzy Subtractive Clustering.** Browsing behavior of users changes dynamically over time. Web usage modeling is static, mainly due to the time complexity of model compilation; adding a new data point may require complete remodeling. Being computationally intensive, remodeling is done occasionally and the models used normally lag behind the current usage patterns leading to irrelevant and mistargeted recommendations. Even though development of maintenance schemes, which adapt these models to non-stationary volatile web, is very important, it has received less attention so far. We propose a web usage profile maintenance scheme using an extension of RFSC called incremental RFSC [Suryavanshi et al. 2005d]. Incremental RFSC can efficiently add new usage data to an existing model without incurring the expense associated with frequent remodeling. We validate our scheme by showing close similarity between complete reclustering and the clustering model obtained after applying our incremental RFSC technique. Any maintenance scheme based on incremental update of the profile requires a measure to indicate, to the web analyst, when accumulated usage data has to be reclustered; otherwise continued maintenance leads to irrelevant, obsolete model. We introduce a quantitative measure, called *impact factor*. When the impact factor exceeds a predefined threshold, remodeling is recommended.

1.5 Outline of Thesis

The rest of this thesis is organized as follows.

In Chapter 2, we present an overview of the web personalization process and the related work in this domain. In Chapter 3, we present our algorithm, Relational Fuzzy Subtractive Clustering (RFSC) and provide a comprehensive experimental study comparing RFSC and Fuzzy C-Means based relational clustering algorithm. In Chapter 4, we show how RFSC models can be used for personalization. We present techniques which are improvements over cluster based and association rule based recommender systems. We conclude this chapter by comparing different collaborative filtering techniques. Our usage profile maintenance scheme is presented in Chapter 5. At the core is another new algorithm, called as the incremental RFSC algorithm. We show correctness of this algorithm, and through experiments, demonstrate its robustness and efficiency for adaptive usage profiling[1]. Chapter 6 includes a summary and conclusions together with some directions for future work.

[1] These results have been submitted as a chapter in the WebKDD book edited by Nasraoui, Zaiane, Spiliopoulou, Mobasher, Masand, and Yu.

Chapter 2

Background and Related Work

In this chapter, we study the personalization process based on web usage mining. A general architecture showing the components of this process is depicted in Fig. 1. We use this as a guide for our exposition of the personalization process in this chapter.

2.1 Collection of Web Data

Data about users browsing the web site can be collected explicitly or implicitly. Explicit data is collected through active involvement of the user, typically from fill-in forms (registration) and questionnaires. Such data may contain generic information such as date of birth and area code, along with some dynamic information, which is likely to change over time such as favorite television programs or football teams. Explicit data collection requires users to exert most of the efforts and make the initial investment, and hence depends much on user's motivation. On the other hand, users are not directly involved in implicit data collection. The navigation behavior and preferences of the user can be learnt typically through analysis of the user accesses and usage of the website. The web server is an important data source for performing web usage mining because it explicitly records the browsing behavior of the visitors to the site. The data recorded in server logs reflects the (possibly concurrent) access of a web site by multiple users. Data collection can also

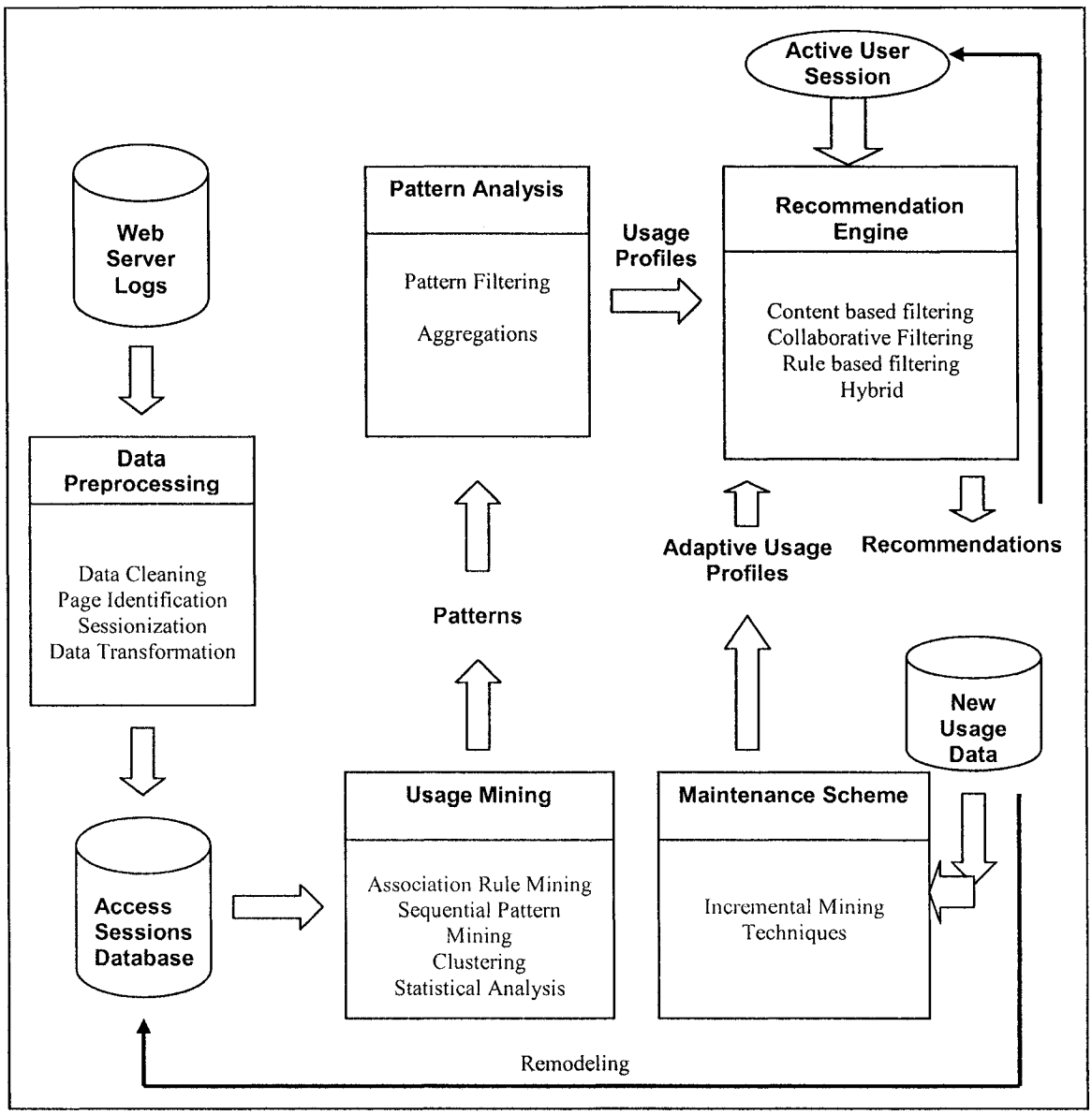


Figure 1 Web Personalization Process

be done at the client side by using a remote agent (such as Javascripts or Java applets) or by modifying the source code of an existing browser (such as Mosaic or Mozilla) to enhance its data collection capabilities. However, these methods require explicit user cooperation, which is not always possible. Another data source is web proxy server, which acts as an intermediate level of caching between client browsers and web servers. This may serve as a good data source for characterizing the browsing behavior of a group

of anonymous users sharing a common proxy server. For more details see [Srivastava et al., 2000].

2.2 Data Preprocessing

Log files produced on the web servers are text files with a row for each HTTP transaction. Fig. 2 shows a typical log file.

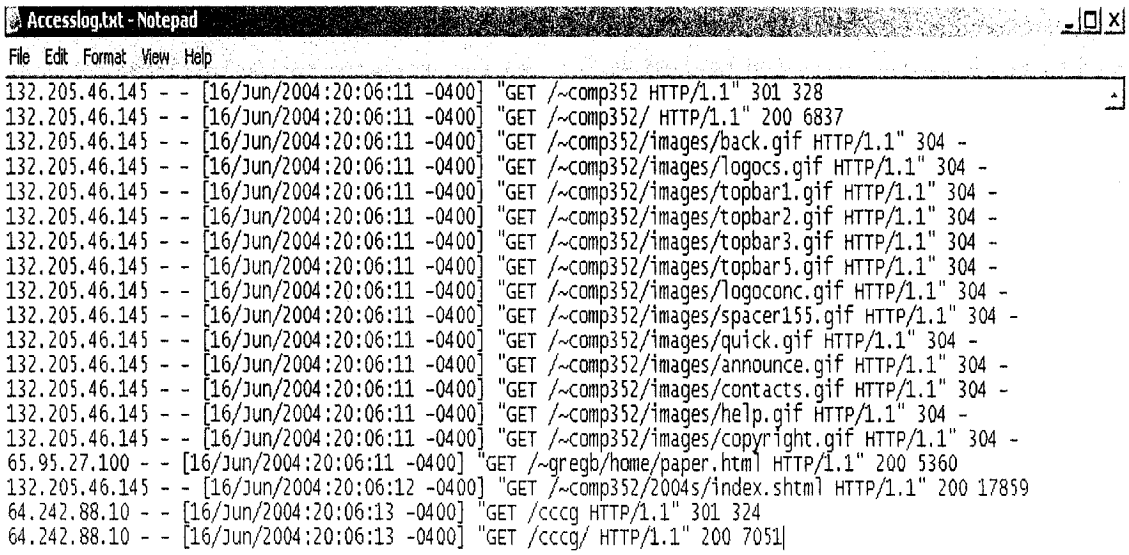


Figure 2 Example of a log file from web server

Let us consider a typical log entry:

195.162.218.155 - - [27/Jun/2004:00:01:54 -0400] "GET /cccg/ HTTP/1.1" 200 38890 "/events/" "Mozilla/4.0 (compatible; MSIE6.0; Windows NT 5.1;SKY11a)"

The first part of this line, 195.162.218.155, specifies the IP-address of the client who made the request to the server. This IP-address can be used to identify different visitors of the site. The second component in this line gives us information on when the request was made. The third part "GET /cccg/ HTTP/1.1" of this entry is called the request line and consists of three parts: The GET-part specifies the method used to request the page. If the

surfer requests a normal web page, then the method used will always be GET. Other possibilities are POST, to send values of a form to the server, and HEAD. The second part of the request line indicates which file was requested. The final part specifies the protocol used to request the file. The two numbers, 200 and 38890, following the request line indicate a status code and the size of the returned file, respectively. The status code 200 indicates that the request was successfully completed. Other status codes indicate various types of errors, from which "error 404: Page not found", is probably the most familiar to the reader. The next part of the line designates the referrer. This is the page that refers to the requested page. From this line, it can thus be concluded that there was a link from "/events/" to the page "/cccg/". Finally, the last component of the line specifies the agent used (browser).

The required high-level tasks in usage data preprocessing include data cleaning, page identification, user identification, session identification (or sessionization), and the inference of missing references due to caching. We briefly discuss some of these tasks below. For a more detailed discussion interested readers are referred to [Cooley, 2000; Cooley et al., 1999, Mobasher 2004].

Data cleaning results in filtering out log entries that are not required or are irrelevant for our task. These include entries that: (i) result in any error (indicated by the error code), (ii) use a request method other than "GET", or (iii) record accesses to image files (.gif, .jpeg, etc), which are typically embedded in other pages and are only transmitted to the user's machine as a by product of the access to a certain web page which has already been logged. Also references due to spiders and web robots are removed. Sessionization is the process of segmenting the access log of each user into sessions. Web sites without

the benefit of additional authentication information from users and without mechanisms such as embedded session IDs must rely on heuristic methods for sessionization. The goal of a sessionization heuristic is reconstruction of the real sessions, where a real session is the actual sequence of activities performed by one user during a visit to the site. Generally, sessionization heuristics are time-oriented where either global or local time-out estimates are applied to distinguish between consecutive sessions.

The preprocessing tasks ultimately result in a set of M pages (URLs), $U = \{url_1, url_2, \ldots, url_M\}$, and a set of N_U user sessions, $S = \{s_1, s_2, \ldots, s_{Nu}\}$, where each $s_i \in S$ is a subset of U. Conceptually, we can view the i^{th} session s_i as a sequence or vector of l pairs

$$s_i = \langle (p_1, w(p_1)), (p_2, w(p_2)), \ldots, (p_l, w(p_l)) \rangle,$$

where each $p_i = url_j$ for some $j \in \{1, \ldots, M\}$, and $w(p_i)$ is the weight associated with each p_i in session s_i. In most web usage mining tasks, we can choose two types of weights: binary, representing the existence or non-existence of a page in the session; or duration of time spent on each page in the session by the user.

Whether the sessions are viewed as sequences or as sets (without considering the order of web page access) depends on the goal of the analysis and the intended applications. For sequence analysis and the discovery of frequent navigational patterns, one must preserve the ordering information in the underlying session. On the other hand, for clustering tasks as well as for collaborative filtering based and association rule discovery, we can represent each user session as an M-dimensional vector, where dimension values are the weights of these pages.

In our work, we use time-oriented heuristic and consider binary weights. Hence, each session is an M-dimensional binary vector such that:

s_{ij} = 1 if the i^{th} session had the j^{th} URL clicked

s_{ij} = 0, otherwise.

In addition to the aforementioned preprocessing steps leading to sessions, there are also a variety of data transformation tasks that can be performed on the session data [Mobasher et al., 2001, 2004].

2.3 Pattern Discovery

We now present specific data mining techniques that are popularly used for mining web usage data. As noted above and depicted in Fig. 1, after applying the various preprocessing steps we ultimately get a set of M pages and N_U sessions. Give a set of sessions, a variety of knowledge discovery and data mining (KDD) [Han and Kamber, 2000] techniques can be applied to obtain actionable patterns. In this thesis, we focus on association rules and clustering.

2.3.1 Association Rules

Association rule mining, one of the most important and well-researched techniques of data mining, was first introduced in [Agrawal and Srikant, 1994]. It aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories. When applied to web usage sessions, association rules are used to find associations among web pages that frequently appear together in that set of sessions. A typical result has the form:

A.html, B.html $\Rightarrow$ C.html

which states that if a user has visited page A.html and page B.html, it is very likely that in the same session the same user has also visited page C.html.

More formally, association rule mining problem can be stated as follows. Let $I = \{I_1, I_2, . . , I_m\}$ be a set of m distinct items, T be a transaction that contains a set of items, that is, $T \subseteq I$, D be a database that is a set of transactions. Association rules are implications of the form $X \Rightarrow Y$, where $X,Y \subset I$ are sets of items, called *itemsets*, and $X \cap Y = \varnothing$. The rule $X \Rightarrow Y$ states that the transactions which contain the items in X are likely to contain also the items in Y. X is also called the antecedent of the rule while Y is its consequent. The two basic parameters of Association Rule Mining (ARM) are: support and confidence.

Support of an association rule is defined as the percentage/fraction of transactions in the database D that contain $X \cup Y$ to the total number of transactions in D. Support is calculated by the following formula:

Support ($X \Rightarrow Y$) = (number of transactions that contains itemsets X and Y) / total number of transactions in the database D.

From the definition, we can see that the support of an item is a statistical significance of an association rule. Suppose the support of an item is 0.1%, it means only 0.1 percent of the transactions in D contain the item. The analyst need not pay much attention to items that are not occurring frequently, obviously a high support is desired for more interesting association rules. Before the mining process, users need to specify the minimum support as a threshold, which means they are only interested in certain association rules that are generated from those itemsets whose supports exceed the given threshold.

Confidence of an association rule is defined as the percentage/fraction of the number of transactions in D that contain $X \cup Y$ to the total number of transactions that contain X,

where if the percentage exceeds the threshold of confidence an interesting association rule $X \Rightarrow Y$ can be generated.

Confidence ($X \Rightarrow Y$) = Support ($X \Rightarrow Y$) / Support (X).

Confidence is a measure of strength of the association rules. Suppose the confidence of the association rule $X \Rightarrow Y$ is 80%. Then it means that 80% of the transactions that contain X also contain Y. As was the case for support, minimum confidence is also pre-defined by the user to ensure interesting rules.

Association rule mining (ARM) is to extract association rules that satisfy the predefined minimum support and confidence from a given database. The problem is usually decomposed into two subproblems. One is to find those itemsets whose occurrences exceed a predefined support threshold in the database. Such itemsets are called *frequent* itemsets. The second problem is to generate association rules from frequent itemsets with the constraints of minimal confidence. Suppose one of the frequent itemsets is L_k, where $L_k = \{I_1, I_2, \ldots, I_{k-1}, I_k\}$. Then association rules with this itemset are generated in the following way: the first rule is $\{I_1, I_2, \ldots, I_{k-1}\} \Rightarrow \{I_k\}$, by checking the confidence this rule can be determined as interesting or not. Other rules can be found in a similar manner. In fact the number of such rules is exponential in the number of items considered. Since the second subproblem is quite straight forward, most of the research is focused on the first subproblem [Han and Kamber, 2000].

2.3.2 Clustering

Clustering is partitioning data into groups of similar objects. Each group, called a cluster, consists of objects that are similar between themselves and dissimilar to objects of other groups. Representing data by fewer clusters necessarily loses certain fine details, but

achieves simplification. The goal of clustering is to separate a finite unlabeled data set into a finite and discrete set of "natural," hidden data structures. Clustering is a subjective process in nature, which precludes an absolute judgment as to the relative efficacy of all clustering techniques [Jain and Dubes, 1988].

There is no universally agreed upon definition of the clustering problem. Here, we give a simple mathematical description based on the descriptions in [Bezdek 1982]. Consider a data set X consisting of N data points (also called objects, instances, cases, patterns, tuples, transactions) where $X= \{x_1, \ldots, x_j, \ldots x_N\}$. Often, but not always, each object can be represented by a set of d measurements; that is, $x_j = (x_{j1}, x_{j2}, \ldots, x_{jd})^T \in \Re^p$ and each measure x_{ji} is said to be a feature (attribute, dimension, or variable). Clustering attempts to seek k-partitions of X, $C=\{C_1, \ldots, C_k\}$ where $k \leq N$, such that

$C_i \neq \phi, i = 1, \ldots, k;$

$\bigcup_{i=1}^{k} C_i = X;$

$C_i \cap C_j = \phi, i,j = 1, \ldots, k \text{ and } i \neq j.$

Cluster analysis typically consist of the four basic steps [Xu and Wunsch, 2005] given below.

1) *Feature selection or extraction.* Feature selection chooses distinguishing features from a set of candidates, while feature extraction utilizes some transformations to generate useful and novel features from the original ones. Both are very crucial to the effectiveness of clustering applications. Elegant selection of features can greatly decrease the workload and simplify the subsequent design process.

2) *Clustering algorithm design or selection.* The step is usually combined with the selection of a corresponding proximity (or distance or dissimilarity) measure and the construction of a criterion function. Patterns are grouped according to whether they resemble each other. Obviously, the proximity measure directly affects the formation of the resulting clusters. Once a proximity measure is chosen, the construction of a clustering criterion function makes the partition of clusters an optimization problem. Clustering is ubiquitous, and a wealth of clustering algorithms has been developed to solve different problems in specific fields. However, there is no clustering algorithm that can be universally used to solve all problems. Therefore, it is important to carefully investigate the characteristics of the problem at hand, in order to select or design an appropriate clustering strategy.

3) *Cluster validation.* Given a data set, each clustering algorithm can always generate a division, no matter whether the structure exists or not. Moreover, different approaches usually lead to different clusters; and even for the same algorithm, parameter identification or the presentation order of input patterns may affect the final results. Therefore, effective evaluation standards and criteria are important to provide the users with a degree of confidence for the clustering results derived from the used algorithms.

4) *Results interpretation.* The ultimate goal of clustering is to provide users with meaningful insights from the original data, so that they can effectively solve the problems encountered. Experts in the relevant fields interpret the data partitions. Further analysis, even experiments, may be required to guarantee the reliability of extracted knowledge.

2.4 Recommendation and Personalization

As introduced in Chapter 1, the recommendation problem is to estimate ratings for the items that have not been seen by a user [Adomavicius and Tuzhilin, 2005]. In web personalization, recommender engines recommend objects in the form of pages, products, advertisements, etc., depending on the type and taste of the user. Also as noted earlier, there are three basic approaches used in recommender systems, namely content-based filtering, rule-based filtering, and collaborative filtering. Collaborative filtering (CF) is the most successful and widely used recommender system technology and is also the approach adopted in our research for web personalization.

More formally, let U denote the set of URLs and M=|U| be the total number of URLs in U. $s_1,\ldots,s_{Nu}$ are the access sessions in the user database D, and Let s_a be the active user session. Let $NA \subset U$ be all the URLs not yet accessed by the active user for which we would like to provide recommendations. A collaborative filter is a function f, defined as shown below, which takes all past users' sessions as the input, and produces recommendation values for pages not yet accessed by the active user [Pennock et al., 2000]:

$$s_{aj} = f(s_1, s_2, \ldots, s_{Nu}), \qquad \text{for all } j \text{ in } NA$$

[Breese et al., 1998] identified two major classes of collaborative filtering algorithms, memory-based and model-based. We discuss these approaches briefly below.

2.4.1 Memory-based CF

Memory-based collaborative filtering typically relies on k-Nearest-Neighbor (kNN) approach comparing the accesses or activities of target user with the historical records of other users in order to find the top k users who have similar tastes or interests. The mapping of a visitor record to its neighborhood could be based on similarity in ratings of

items, access to similar content or pages, or purchase of similar items. The identified neighborhood is then used to recommend items not already accessed or purchased by the active user. In the context of personalization, memory-based CF involves measuring the similarity or correlation between the active session s_a and each session s_j in D. The top k most similar sessions to s_a are considered to be its neighborhood, which is commonly denoted as $NB(s_a)$ [Mobasher, 2004]. A variety of similarity measures can be used to find the nearest neighbors. In traditional collaborative filtering domains (where feature weights are item ratings on a discrete scale), the Pearson correlation coefficient is commonly used [Breese et al., 1998, Sarwar et al., 2000b]. This measure is based on the deviations of users' ratings on various items from their mean ratings on all rated items. However, this measure may not be appropriate when the primary data source is clickstream data (particularly in the case of binary weights). Instead the cosine coefficient, commonly used in information retrieval, which measures the cosine of the angle between two vectors, can be used. The cosine coefficient can be computed by normalizing the dot product of two vectors with respect to their vector norms. Hence cosine similarity between sessions s_a and s_j is:

$$\text{sim}(s_a, s_j) = \text{cosine}_{a,j} = \frac{\sum_k (s_{ak} \cdot s_{jk})}{\sqrt{\sum_k s_{ak}^2 \sum_k s_{jk}^2}};$$

In order to determine which items (not already visited by the user in the active session) are to be recommended, a recommendation score is computed for each item or page *p* based on the neighborhood for the active session. Two factors are used in determining this recommendation score: the overall similarity of the active session to the neighborhood as a whole, and the average weight of each item in the neighborhood.

First we compute the mean vector (centroid) of NB(s_a), i.e., popularity of each item or page among the neighbors. Popularity of each page in the mean vector is computed by finding the ratio of the sum of the page weights across sessions to the total number of sessions in the neighborhood. Let this centroid be denoted as cent(NB(s_a)). For each page p in the neighborhood centroid, we can now obtain a recommendation score as a function of the similarity of s_a to the centroid vector cent(NB(s_a)) and the weight of p in cent(NB(s_a)). Hence the recommendation score [Mobasher, 2004] of page p denoted as rec(s_a , p) is:

$$\text{rec}(s_a, p) = \sqrt{weight(p, NB(s_a)) \times sim(s_a, cent(NB(s_a)))}$$

where weight(p, NB(s_a)) is the weight of p in cent(NB(s_a)).

If p is in the current active session, then its recommendation value is set to zero. If a fixed number N of recommendations are desired, then the top N items with the highest recommendation scores are considered to be part of the recommendation set. Alternately, we could also recommend only those pages whose recommendation score is above a threshold.

2.4.2 Model-based CF

In contrast to memory-based methods, model-based algorithms [Billsus and Pazzani, 1998, Chien and George, 1999, Getoor and Sahami, 1999, Breese et al. 1999, Pennock and Horvitz, 1999], use the historic rating or dataset to learn a model, which is then used to make rating predictions. Model-based CF algorithms typically depend on the underlying modeling technique that is used. There are several model-based collaborative recommendation approaches proposed in the collaborative filtering and recommender system literature, for details see [Adomavicius and Tuzhilin, 2005]. A statistical model

for collaborative filtering was proposed in [Ungar and Foster, 1998], and several different algorithms for estimating the model parameters were compared, including K-means clustering and Gibbs sampling. Other collaborative filtering methods include a Bayesian model [Chien and George, 1999], a probabilistic relational model [Getoor and Sahami, 1999], and a linear regression [Sarwar et al., 2001]. [Shani et al., 2002] view the recommendation process as a sequential decision problem and propose using Markov decision processes for generating recommendations. A different approach to improving the performance of existing collaborative filtering algorithms was taken in [Yu et al., 2002], where the input set of user-specified ratings is carefully selected using several techniques that exclude noise, redundancy, and exploit the sparsity of the ratings' data.

In the domain of web personalization, one of the most advanced systems is the WebPersonalizer, proposed by Mobasher et al. [1999, 2000]. WebPersonalizer provides a framework for mining web log files to discover knowledge for making recommendations to current users based on their browsing similarities to previous users. Data mining techniques such as association rules, sequential pattern discovery, clustering, and classification are applied, in order to discover interesting usage patterns. [Perkowitz and Etzioni 1998] were the first to define the notion of adaptive web sites as sites that semi automatically improve their organization and presentation by learning from visitor access patterns. The authors propose PageGather, an algorithm that uses a clustering methodology to discover web pages visited together and to place them in the same group. In a more recent work [Perkowitz and Etzioni 2000], they move from the statistical cluster-mining algorithm PageGather to IndexFinder, which fuses statistical and logical information to synthesize index pages. [Nasraoui et al. 2000, Nasraoui et al. 2002]

introduce the notion of uncertainty in web usage mining, discovering clusters of user session profiles using fuzzy algorithms. In their approach, a user or a page can be assigned to more than one cluster. They introduce a similarity measure that takes into account both the individual URLs in a Web session, as well as the structure of the site. They also present a personalization system that uses these mined profiles to recommend pages.

[Masseglia et al. 1999a,b] apply data mining techniques such as association rules and sequential pattern discovery on Web log files and then use them to customize the server hypertext organization dynamically. The prototype system, WebTool, provides a visual query language in order to improve the mining process. [Shahabi et al. 2001, 2003] developed a recommendation system, termed Yoda that is designed to support large-scale web-based applications requiring highly accurate recommendations in real-time. With Yoda, they introduce a hybrid approach that combines collaborative filtering and content-based querying to achieve higher accuracy. It uses clustering for model learning. [Coenen et al., 2000] propose a framework for self-adaptive Web sites, taking into account the site structure except for the site usage. The proposed approach is based on the fact that the methods used in web usage mining produce recommendations including links that don't exist in the original site structure, resulting in the violation of the beliefs of the site designer and the possibility of making the visitor get lost by following conceptual and not real links. Therefore, they suggest that any strategic adaptations based on the discovery of frequent item sets, sequences, and clusters should be made offline and the site structure should be revised.

Chapter 3

Relational Fuzzy Subtractive Clustering

As discussed in Chapter 2, clustering is partitioning of a collection of objects into disjoint subsets or clusters such that objects in the same cluster have some common properties that distinguish them from objects in other clusters. A fuzzy clustering is a generalization of this process, where the clusters are not necessarily subsets of the collection, but instead they are fuzzy subsets, based on the notion of fuzzy sets introduced by [Zadeh 1965]. That is, each object is assigned a number between 0 and 1 with respect to every cluster, called as *grade of membership*. Objects which are similar to each other are identified by having high memberships in the same cluster. "Hard" clustering algorithms assign each object to a single cluster that is using the two distinct membership values of 0 and 1. In many situations, say, web usage data[2], this "all or none" or "black or white" membership restriction is not realistic as very often there may not be sharp boundaries between clusters and many objects may have characteristics of different classes with varying

[2] Web log data is inherently soft and fuzzy in nature. The browsing behavior of users on the web is highly uncertain. A user might browse the same page for different purposes. Each time the user accesses the site, he/she may have different browsing goals. The same user in the same session may have different sub-goals and interests. It is therefore inappropriate to capture such overlapping interests of the users in crisp partitions.

degrees. In such situations, it is more natural to assign to each object a set of memberships, one for each cluster or class. The implication of this is that the class boundaries are not hard but rather fuzzy. The main advantage of fuzzy clustering over hard clustering is that it yields more detailed information about the underlying structure of the data.

Let us formally distinguish between a hard and fuzzy clustering. Consider a data set X consisting of N data points where $X= \{x_1, \ldots, x_j, \ldots x_N\}$ is a subset in feature space $\Re^p$. Let C be an integer such that $2 \leq C < N$. A hard C-partition of X can be represented by a (C×N) matrix $U = [u_{ik}]$ such that,

1) $u_{ik} \in \{0, 1\}; \qquad 1 \leq i \leq C; 1 \leq k \leq N,$

2) $\sum_{i=1}^{C} u_{ik} = 1; \qquad 1 \leq k \leq N,$

3) $\sum_{k=1}^{N} u_{ik} > 0; \qquad 1 \leq i \leq C.$

A fuzzy C-partition of X is also represented by a (C×N) matrix $U= [u_{ik}]$ such that,

1) $u_{ik} \in [0, 1]; \qquad 1 \leq i \leq C; 1 \leq k \leq N,$

2) $\sum_{i=1}^{C} u_{ik} = 1; \qquad 1 \leq k \leq N,$

3) $\sum_{k=1}^{N} u_{ik} > 0; \qquad 1 \leq i \leq C.$

U is called the membership matrix, and u_{ik} is the membership grade of k^{th} object to i^{th} cluster. The first condition for hard clustering means that each object either belongs to a cluster or it does not, i.e., u_{ik} is either 0 or 1. While for fuzzy clustering it means each object can belong to a cluster with varying degree of memberships in [0, 1]. We also see

that conditions 2 and 3 essentially remain the same for both. Condition 2 can be interpreted differently. For hard clustering it means that each object belongs to exactly one cluster while for fuzzy clustering the sum of memberships of an object to different clusters is 1. Also condition 3 means that no cluster can be empty.

To deal with the fuzziness and uncertainty in web usage data, [Nasouri et al, 2000] proposed to extract profiles using an unsupervised relational clustering algorithm based on the competitive agglomeration algorithm. They [Nasraoui et al. 2002] further extend this approach with fuzzy clustering algorithms such as Relational Fuzzy C-Maximal Density Estimator (RFC-MDE) and Fuzzy C Medoids algorithm (FCMdd). The basis of these techniques is the Fuzzy C-means (FCM) [Bezdek, 1982] method of clustering which allows one piece of data to belong to two or more clusters. We next describe relational data on which all the aforementioned techniques work.

3.1 Relational Data

The given set of objects to be clustered is often described using either numerical object data or numerical relational data [Hathway et al., 1996]. Numerical object data is available in the form of a set of vectors in $\Re^p$, say $X= \{x_1, \ldots, x_j, \ldots x_N\}$ where x_j is feature vector representing object j and $x_{jk} \in \Re^p$ is the k^{th} feature of x_j, and $\Re^p$ is feature space. Algorithms that generate partitions (hard or fuzzy) of object data are usually called *object clustering algorithms*. Relational data refers to the situation where we have only numerical values representing the degree of similarity or dissimilarity (relation) between the pair of objects to be clustered. Relational data may be based on actual object data or can be based on subjective expert knowledge. Algorithms that generate partitions of relational data are called as *relational clustering algorithms*. Relational clustering is more

general in the sense that it is applicable in situations in which the objects to be clustered cannot be represented by numerical features. Relational data is typically represented by a matrix R, where R_{ij} is the dissimilarity between the i^{th} and the j^{th} object. It holds that $R_{ij} \geq 0$, $R_{ij}=R_{ji}$, and $R_{ii}=0$. Further, R may be a Euclidean or non-Euclidean matrix. R is said to be Euclidean if there exist a set of N object data points in some p-space whose squared Euclidean distances match the values in R. More formally, relation $R= [r_{ij}]$ is Euclidean if there exists a data set $X= \{x_1, \ldots, x_j, \ldots x_N\}$ in $\Re^p$, such that $r_{ij} = \|x_i - x_j\|$ where $\| \cdot \|$ is the Euclidean norm.

Object clustering algorithms are not suitable for clustering access sessions [Nasraoui et al., 2000]. This is because of the high dimensionality of the feature space (there are usually thousands of URLs in a typical web site) and the likely correlation between features. The web sessions are too complex to convert to simple numerical features, particularly because the organization of the web site must be taken into account. In fact, the URLs in a site have a hierarchical or tree-like structural composition. Therefore when defining the relation R for sessions, the similarity or dissimilarity measure between sessions should take into account both the structure of the site as well as the URLs involved. Such a relational matrix of access sessions is essentially non-Euclidean in nature.

β-spread transformations

The aforementioned algorithms based on FCM such as RFC-MDE and FCMdd as well as Non-Euclidean Relational Fuzzy clustering (NERF) [Hathaway and Bezdek, 1994] convert non-Euclidean relations into Euclidean using the β-spread transformations. This transformation consists of adding a positive number β to all off-diagonal elements of the

relational matrix R. It has been shown [Hathaway and Bezdek, 1994] that there exists a positive number β_0 such that the β spread transformed relation R_β is Euclidean for all $\beta \geq \beta_0$, and is non Euclidean for all $\beta < \beta_0$. The value β determines the amount of spreading, and hence should be chosen as small as possible to avoid unnecessary spreads of data which may lead to loss of cluster information. On the other hand, exact computation of β_0 involves expensive eigenvalue computations [Corsini et al, 2004]. [Hathaway and Bezdek, 1994] show that β-spread transformation can be computed dynamically during the iteration process of FCM. The β_N computed is the minimum value, which guarantees convergence of the computation. However, the performance of algorithms using this transformation depends on the value of β_N which could be so large that the structure in the original relational matrix R might not be mirrored by that in $R_{\beta N}$ [Corsini et al., 2004]. In a recent development, this requirement of a computationally expensive β-spread transformation has been overcome with the introduction of Any Relational Clustering algorithm (ARCA) proposed in [Corsini et al. 2004]. ARCA is the most recent evolution of FCM based algorithms, and is an appropriate candidate from this class of clustering techniques when carrying out comparative experimental studies of the kind we have discussed later in this chapter. ARCA is discussed in more detail below.

3.2 Any Relational Clustering Algorithm

ARCA takes the FCM algorithm as starting point. FCM is an iterative algorithm, which partitions a dataset by minimizing the Euclidean distance between each point belonging to a cluster and the prototype of the cluster. In FCM, a prototype is a point (possibly not included in the original dataset), which is representative of the cluster and has a position

around the center of the cluster with respect to the neighboring objects. This resulting cluster is obtained by updating at each iteration, both the membership of each point to a cluster and the cluster prototypes.

FCM determines a fuzzy partition of the dataset using an alternating optimization scheme to iteratively minimize appropriate objective functions J_m defined as follows:

$$J_m(U,V) = \sum_{i=1}^{C}\sum_{k=1}^{N_U} {u_{ik}}^m d^2(x_k, v_i)$$

under the constraints, $\forall$i, k, $u_{ik} \in [0,1]$, and $\forall$k, $\sum_{i=1}^{C} u_{ik} = 1$

where $x_1, x_2, \ldots, x_{Nu}$ are the objects to be clustered,

v_i is the centroid (prototype) of cluster i,

u_{ik} is the membership degree of object x_k to the cluster represented by v_i,

$d(x_k, v_i)$ is the Euclidian distance between i^{th} centroid (v_i) and k^{th} data point, and

$m \in [1,\infty]$ is a weighting exponent (fuzzifier).

The membership u_{ik} and the cluster centers v_i is updated at each iteration as follows:

$$v_i = \frac{\sum_{k=1}^{N_U} {u_{ik}}^m x_k}{\sum_{k=1}^{N_U} {u_{ik}}^m}; \qquad u_{ik} = \frac{1}{\sum_{j=1}^{C}\left(\frac{d_{ik}}{d_{jk}}\right)^{2/(m-1)}};$$

In ARCA, each object is represented by the vector of its relation strengths with other objects in the dataset, and a prototype (v_i) is an object (possibly not included in the original data set) whose relationship with all the objects in the dataset is representative of the mutual relationships of a group of similar objects. As in FCM, ARCA partitions the

dataset by minimizing the distance between each object (strongly) belonging to a cluster and the prototype of the cluster. ARCA yields an optimal partition by minimizing the following objective function:

$$J_m(U,V) = \sum_{i=1}^{C}\sum_{k=1}^{N_U} {u_{ik}}^m \delta^2(x_k, v_i)$$

subject to the conditions, $\forall$i, k, $u_{ik} \in [0,1]$, and $\forall$k, $\sum_{i=1}^{C} u_{ik} = 1$

with similar notations defined above in the FCM objective function.

Here, $\delta(x_k, v_i)$ is the deviation of the relation between x_k and all the other objects, and between v_i and all the other objects, defined as:

$$\delta(x_k, v_i) = \sqrt{\sum_{j=1}^{N_U} (R_{kj} - v_{ij})^2}$$

where R_{kj} is the relation between objects x_k and x_j, and v_{ij} is the relation between prototype v_i and object x_j. We now present the ARCA algorithm.

ARCA algorithm

Fix C and m, where $2 \le C \le N_U$ and $1 < m < \infty$, and choose an initial partition $U(0) = [u_{ik}^{(0)}]$.

For $l = 0, 1, 2, \ldots$

1) Determine the prototype vector $V^{(l)} = [v_{ik}^{(l)}]$ as:

$$v_{ik}^{(l)} = \frac{\sum_{j=1}^{N_U} {u_{ij}}^m R_{kj}}{\sum_{j=1}^{N_U} {u_{ij}}^m};$$

2) Update $U(l)$ to $U(l+1)$ using the formula

$$u_{ik}^{(l+1)} = \frac{1}{\sum_{j=1}^{C} \left(\frac{\delta_{ik}^{(l)}}{\delta_{jk}^{(l)}} \right)^{2/(m-1)}};$$

3) Compare U(l) and U(l+1) in a convenient matrix norm: if $\| U(l+1) - U(l) \| < \varepsilon$, where $\varepsilon > 0$ is a prefixed termination threshold, then stop; otherwise, set $l = l+1$ and go to step1.

Input Parameters and Constraints

With the above background of FCM and ARCA we now discuss the limitations of these techniques particularly for mining usage data. The success of FCM based algorithms depends on some "carefully" selected user specified control parameters (cases 1 to 3 below) or on the constraint (case 4), as discussed below.

1. *The number of clusters C*: This user input parameter is a very critical factor that determines the quality of the clustering. When the dataset is small, it maybe possible to estimate C. However, in case the dataset is huge, as is the case with web log records, it is not known how to estimate C correctly. Thus it is difficult to guarantee that the clustering results would manifest the true structure in the data. One way to decide on the value of C is using a validity index, for which one could use popular validity measures such as Xie-Beni index (S) [Xie and Beni, 1991] and partition coefficient [Bezdek 1982]. The Xie-Beni index depends on the compactness and the separation of the clusters found. The index value S tends to monotonically decrease when C approaches N_U (the number of data objects to be clustered). Hence they suggest to cluster the data from C=2 to $N_U/3$ and then select a value C for which the Xie-Beni index is minimum. This procedure could be impractical when N_U is large as in the case with sessions. For instance, when there are 10,000 sessions to be clustered, the algorithm must be executed for C = 2 to 3333 to determine an optimal C. Further, web analysts may need to do frequent remodeling to capture current trends and interests. The total computation time is prohibitively large

making these algorithms poorly suited for large usage datasets using standard computing resources.

2. *The fuzzifier m*: This parameter controls the extent of membership among fuzzy clusters in the dataset. In general the larger m is, "fuzzier" is the resulting partition i.e. membership assignments; conversely as m approaches 1, the clustering becomes hard. This parameter is also required to be specified for all FCM based techniques. Once again, for large web usage datasets, it is unclear how to specify a value of m *a priori*. The web analyst may have no idea as to "how fuzzy" is the current usage data.

3. *Initial partition U(0) (for RFC-MDE, NERF and ARCA) and initial medoids (for FCMdd)*: An initial partition or medoid must be specified as the start point for the above clustering algorithms. Different choices of this initial partition or medoids can lead to different local extrema of the objective function [Abraham, 2003, Nasraoui et al. 2002].

4. *Partition of unity constraint for membership*: A major restriction of FCM based algorithms is that they require the total sum of the memberships of every object to all the C clusters should be 1,

$$\text{i.e., } \sum_{i=1}^{C} u_{ik} = 1; \qquad 1 \le k \le N_U,$$

A careful study of this formulation shows that it may cause bias due to noise. Hence, just to be able to satisfy this constraint, even outliers will get assigned to one or more classes with some degree of membership. Almost every dataset has some noise and more so the usage data which is inherently noisy in nature. To overcome this restriction, [Dave 1991] introduced the concept of noise clustering (NC) algorithm for object data, and suggested considering an extra cluster in addition to the C clusters specified by the user. This noise

cluster is represented by a prototype that has the same distance δ from all the feature vectors. Later on NC was extended to relational data; however selection of δ is a complex issue [Dave and Sen, 2002]. It requires a good knowledge of the data or a good estimate of it, which could be rather difficult to obtain when the data is large.

3.3 Relational Fuzzy Subtractive Clustering Algorithm

We now present our new technique for clustering web usage data which we have named as *Relational Fuzzy Subtractive Clustering* (RFSC) [Suryavanshi et al., 2005a]. As the name suggests, RFSC is based on the subtractive clustering algorithm proposed in [Chiu 1994], a technique widely used in fuzzy systems modeling. It is important to mention here that RFSC overcomes all the above-mentioned shortcomings. .

[Yager and Filev 1992] proposed a simple and effective algorithm, called the mountain method, for estimating the number and initial location of cluster centers. Their method is based on gridding the data space and computing a potential value for each grid point based on its distances to the actual data points; a grid point with many data points nearby will have a high potential value. The grid point with the highest potential value is chosen as the first cluster center. The key idea in their method is that once the first cluster center is chosen, the potential of all grid points is reduced according to their distance from the cluster center. Grid points near the first cluster center will have greatly reduced potential. The next cluster center is then placed at the grid point with the highest remaining potential value. This procedure of acquiring new cluster center and reducing the potential

of surrounding grid points repeats until the potential of all grid points falls below a threshold. Although this method is simple and effective, the computation cost grows exponentially as the dimension of the problem increases. For example, a clustering problem with 4 variables and each dimension having a resolution of 10 grid lines would result in 10^4 grid points that must be evaluated. [Chiu, 1994] proposed an extension of mountain method, called subtractive clustering, in which each data point, not a grid point, is considered as a potential cluster center. The main advantage of this method is that it eliminates the need to specify a grid resolution, in which tradeoffs between accuracy and computational complexity must be considered. RFSC uses the same idea of considering each data object as a potential cluster center. It uses a potential function for relational data which is derived on the same lines as the one used in [Chiu, 1994]. We describe the RFSC algorithm in detail below.

The potential P_i of any object x_i is calculated using the function:

$$P_i = \sum_{j=1}^{N_U} e^{-\alpha R_{ij}^2}, \text{ where } \alpha = 4/\gamma^2$$

in which R_{ij} is the dissimilarity between objects x_i and x_j, N_U is the number of objects to be clustered, and γ is essentially the neighborhood calculated from the relational matrix R. We define the notion of *neighborhood-dissimilarity* (γ_i) of each object x_i from every other object as the median of dissimilarity values of x_i to all other objects. The neighborhood-dissimilarity value γ for the entire dataset is defined as the median of all γ_i's, for $1 \leq i \leq N_U$. Further, for RFSC algorithm we impose a restriction of having normalized dissimilarity values, i.e., the relational matrix R is required to be a dissimilarity matrix such that $0 \leq R_{ij} \leq 1$. Therefore, γ will always be a value in the range

[0, 1]. This is a heuristic that seems to work fine for many datasets which we have used in our numerous experiments.

The object with the highest potential (P_1*) is selected as the first cluster center. Next, the potential of each object is reduced proportional to the degree of similarity with this previous cluster center. Thus, there is larger subtraction in potential of objects that are closer to this cluster center compared to those which are farther away. After this subtractive step, the object (x_t) with the next highest potential (P_t) is selected as the next candidate cluster center. Now to decide whether this candidate cluster center can be accepted as an actual cluster center or should be rejected, we make use of two threshold values, called *accept ratio* (denoted as $\overline{\in}$) and *reject ratio* ($\underline{\in}$) such that $0 < \underline{\in}, \overline{\in} < 1$, and $\underline{\in} < \overline{\in}$. If $P_t > \overline{\in} P_1$*, then x_t is selected as the next cluster center, and this is followed by the subtractive step described above. If $P_t < \underline{\in} P_1$*, then x_t is rejected, and the clustering algorithm terminates. If the potential P_t lies between $\overline{\in} P_1$* and $\underline{\in} P_1$*, then we say that the potential has fallen in the gray region. In this case, we check if the object provides a good trade-off between having a sufficient potential and being sufficiently far from existing cluster centers. If this is the case, then it is selected as the next cluster center. This process of subtraction and selection continues until $P_t < \underline{\in} P_1$*, which is the termination condition. After finding the cluster centers, say C in number, we can find the membership of different x_j with each cluster c_i using the formula:

$$u_{ij} = e^{-\alpha R_{c_i j}^2}, \; i = [1..C] \; \& \; j = [1..N_U],$$

in which $R_{c_i j}$ is the dissimilarity of the i^{th} cluster center x_{c_i} with the j^{th} session x_j. When $x_j = x_{c_i}$, we have $R_{c_i j} = 0$ and the membership $u_{ij} = 1$. Also in the above process we have

relaxed the constraint which FCM based algorithms, namely that $\sum_{i=1}^{C} u_{ij} = 1$. This effectively makes RFSC less sensitive to noise, as no object is forced any more to have a membership spread in such a way that the sum is 1. Further membership spread of object having characteristics of two or more classes also may not sum up to one. If we carefully observe the RFSC membership function, we see that it is Gaussian in nature and membership is calculated for every cluster using this function. With respect to individual clusters, the objects along the asymptotes of this Gaussian function are noise for that cluster. Noise objects with respect to the entire dataset are therefore essentially defined as objects which lie along the asymptotes of the Gaussian function for all clusters. We also show this immunity to noise property of RFSC through experimental analysis in section 3.4.

RFSC Algorithm

Given a dissimilarity matrix R for N_U objects, we first compute γ as defined above and use it in the RFSC algorithm described below.

1) Set $\overline{\in}$ (accept ratio) and $\in$ (reject ratio);
2) Calculate the potential of each object as follows:

$$P_i = \sum_{j=1}^{N_U} e^{-\alpha R_{ij}^2}, \text{ where } \alpha = 4/\gamma^2$$

Select x_i with the maximum potential as the first cluster center. Set cluster counter $k = 1$, and let $c_k = c_1 = i$. Here, x_{c_k} is the cluster center. Let $P_k^* = P_1^* = P_i$.

3) Revise the potential of each object as follows (subtractive step):

$$P_i = P_i - P_k^* e^{-\alpha R_{ic_k}^2}, \forall i = 1..N_U$$

Hence the potential of object x_{c_k}, which is the k^{th} cluster center, will become zero and all the objects which are similar (close) to this object will also have a greater reduction in their potentials because of this subtraction.
Now, select x_t with the current maximum potential P_t as the candidate for the next cluster center.

4) (Accept):

if $P_t > \overline{\in} P_1^*$, **then** accept x_t as the new cluster center,
increment the cluster counter, $k = k+1$, set $c_k = t$, and go to step 3.

(Reject):

else if $P_t < \underline{\in} P_1^*$, **then** reject x_t and terminate.

else (Gray Area)

Let d_{min} = minimum of the dissimilarity values between x_t and all the previously found cluster centers.

(Accept):

if $(d_{min}/\gamma)+(P_t/P_1^*) \geq 1$, **then** accept x_t as the new cluster center, k= k+1, $c_k = t$, and go to step 3.

(Reject):

else reject x_t as a cluster center and set its potential P_t to 0; Select the object with the next highest potential as the new candidate cluster center, and go to step 3.

endif

endif

endif

3.4 Cluster Validity

Every clustering technique should have a cluster validity index that gives a measure of the goodness or quality of the clustering. More formally, a *validity functional* is a function which assigns to the output of a clustering algorithm a number indicating how well has it identified the structure present in the data. Good clustering of the objects results in least intra-cluster distance and large inter-cluster distance. We define a goodness index for RFSC based on the popularly used Xie-Beni index [Xei and Beni 1991]. Xie-Beni index is basically the ratio of compactness to the separation of the clusters defined for FCM based clustering algorithms that works when we have the constraint $\sum_{i=1}^{C} u_{ij} = 1$, $\forall$ j = 1 to N_U. Since this condition is relaxed in the case of RFSC, we have formulated our own validity index for RFSC clusters by suitably adapting the Xie-Beni formulation. This is described further below.

After applying RFSC for clustering the collection of objects $\{x_1, x_2, \ldots, x_{Nu}\}$ through R, we obtain a set of C cluster centers or cluster prototypes $W= \{Z_1,Z_2,\ldots,Z_C\}$ and the membership grade of each object to C clusters is stored in the matrix u ($C \times N_U$).

Similar to [Xie and Beni, 1991] we define the notion of *fuzzy deviation* of an object x_j to cluster c_i as, $d_{ij} = u_{ij} * R_{c_i j}$; where u_{ij} is the membership of x_j to cluster c_i, and $R_{c_i j}$ is the dissimilarity between i^{th} cluster center and x_j.

For each cluster c_i, the *fuzzy variation* of c_i is the sum of the squares of fuzzy deviations of each object. Thus, $\sigma_i = \sum_{j=1}^{N_U} d_{ij}^2 = \sum_{j=1}^{N_U} u_{ij}^2 * R_{c_i j}^2$;

The total variation σ is the sum of all σ_i, for all i in [1.. C]. The value σ essentially depends on the fuzzy membership grades of different objects to different clusters and the cluster centers themselves. A better partition should result in smaller σ.

The *fuzzy cardinality* of a cluster c_i is defined as, $n_i = \sum_{j=1}^{N_U} u_{ij}$;

For hard partitioning, n_i will be equal to the number of objects in c_i.

The ratio of fuzzy variation of each cluster c_i to fuzzy cardinality of that cluster is called *compactness* of that cluster, and is defined as, $\Pi_i = \sigma_i / n_i$

Total compactness of the clustering normalized over C for RFSC is defined as:

$$\Pi = \frac{1}{C} \sum_{i=1}^{C} \left[\frac{\sum_{j=1}^{N_U} u_{ij}^2 * R_{c_i j}^2}{\sum_{j=1}^{N_U} u_{ij}} \right].$$

The more compact the clusters are, the smaller Π would be.

Separation is defined as min i ≠ k $R^2_{c_i c_k}$, for i = 1 to C and k = 1 to C, where x_{c_i} and x_{c_k} are the i[th] and k[th] cluster centers and $R_{c_i c_k}$ is the dissimilarity between these cluster centers, i.e., the minimum of the distances between any pair of cluster centers found.

$$\text{Index of goodness} = \frac{Compactness}{Separation}$$

During the RFSC clustering based model building process, we search and choose those values of accept ratio ∈ and reject ratio ∈ for which this index is minimum.

3.5 Experiments and Results

We first describe the usage data that we used in our experiments. We follow this by a brief description of the similarity measures we used followed by the study of usage profiles extracted through RFSC. We then present a comprehensive performance study comparing RFSC and ARCA.

3.5.1 Datasets

Over the course of this thesis, we experimented with a number of synthetic datasets and two web usage datasets. We used the access logs from the web server of Computer Science and Software Engineering Department (CSE) at Concordia University during the period of June 15, 2004 to July 5, 2004 (first dataset) and again during the period of December 31, 2004 to March 05, 2005 (a much larger dataset). In this thesis, we only

present the experimental results for the latter dataset[3]. We performed data preprocessing and preparation activities as explained in section 2.2. Log entries for image files and other such components within a web page were removed. Log records for accesses by web crawlers and failed requests are also removed from the dataset. After the cleaning step, we proceed to extract the access sessions. Each distinct URL in the site is assigned a unique number j ranging from 1 to M, the total number of URLs. The collection of the user accesses in the k^{th} session is represented as a binary vector of size M, in which the j^{th} entry is 1 if the user accessed the jth URL during this session, and is 0 otherwise. Sessions were identified from the log records by considering the time heuristic of 45 minutes as the maximum elapsed time between two consecutive accesses from a single IP address. Root "/" was filtered out from all sessions as it appeared in more than 80% of the sessions. We also removed short sessions of length 1 or 2, as they do not carry much information related to users' access patterns. After this pre-processing phase, we obtained 64,529 user sessions with 10,424 distinct URLs. Compared to a few hundred or thousand sessions used for verifying other fuzzy clustering techniques [Nasraoui et al., 2000, 2002], in our work, the numbers of sessions as well as the pages are much larger. The average length of the sessions was 6.997 pages and sparsity level was 0.999329, defined as 1- (nonzero entries / total entries).

3.5.2 Similarity Measures

We use the similarity measure proposed in [Nasraoui et al., 2000, Nasraoui et al., 2002] for construction of relational matrix R. This similarity measure effectively captures the

[3] Experimental results from the former dataset have been reported in our published papers [Suryavanshi et. al., 2005a,b,c,d].

organization or structure of the web site and is popularly used. Session similarity is defined based on URL similarity. The syntactic similarity between the i^{th} and j^{th} URLs is defined as:

$$S_u(i,j) = \min\left(1, \frac{|\, p_i \cap p_j \,|}{\max(1, \max(|\, p_i \,|, |\, p_j \,|) - 1)}\right),$$

where p_i denotes the path traversed from the root node (the main page in the site) to the node which corresponds to the i^{th} URL. The length of path p_i is denoted as $|\, p_i \,|$.

To determine similarity of any two sessions s_k and s_l, two measures may be used. The first measure is cosine, which does not consider the site structure, and is defined as:

$$S_{1,kl} = \sum_{i=1}^{M} s_{ki} s_{li} \Big/ \sqrt{\sum_{i=1}^{M} s_{ki} \sum_{i=1}^{M} s_{li}}$$

The second similarity measure, defined below, incorporates syntactic URL similarity.

$$S_{2,kl} = \sum_{i=1}^{M} \sum_{j=1}^{M} s_{ki} s_{lj} S_u(i,j) \Big/ \sum_{i=1}^{M} s_{ki} \sum_{j=1}^{M} s_{lj}$$

The similarity between any pair of sessions s_k and s_l is a value in the range [0, 1] defined as:

$$S_{kl} = \max\,(S_{1,kl}, S_{2,kl})$$

It follows that the dissimilarity between sessions k and *l* is $D_{kl} = 1 - S_{kl}$, which is a non-Euclidean similarity measure. This means that if FCM based algorithms are to be used for clustering then they would need to convert this non-Euclidean relation into Euclidean through the expensive β spread transformations discussed earlier in Section 3.3.

3.5.3 Usage Profiling through RFSC

After construction of the relation R we can now apply RFSC algorithm to extract usage profiles. RFSC finds clusters in decreasing order of significance i.e. the cluster which is obtained first is the most significant or prominent interest area followed by clusters of lower prominence. We use the index of goodness defined in Section 3.4 to obtain the best clustering. We obtain clustering for values of $\overline{\in}$ and $\underline{\in}$ ranging from 0.1 to 1 in steps of 0.1, such that $\underline{\in} < \overline{\in}$ and find the cluster index of goodness for each combination. Then we choose that combination of $\overline{\in}$ and $\underline{\in}$ for which this index is the best. Hence the index of goodness validates the clusters and is used to obtain the optimal values of $\overline{\in}$ and $\underline{\in}$. We also note that even if we do not chose optimal values of $\overline{\in}$ and $\underline{\in}$, the significant clusters or prominent interest areas will always be found by RFSC. For our data set, the optimal values were found at $\overline{\in}$=0.4 and $\underline{\in}$=0.1 and number of clusters C=46. To display these usage profiles we performed defuzzification of the clustering obtained from RFSC. As a defuzzification step, which is usually required for interpretation of results, each session is assigned to the cluster to which its degree of membership is the highest. In this defuzzification process noise sessions which have low membership with all the clusters are separated and put into an extra cluster, called *noise* cluster. This noise profile is shown in the Table 1.

Usage Profile (UP) for the i^{th} cluster is an M – tuple $UP_i = (up_{i1},..., up_{iM})$, where M is total number of URLs and up_{ij} is the sum of all the sessions that belong to cluster i having the j^{th} URL clicked divided by the sum of all the sessions in cluster i (cluster cardinality). Hence up_{ij} indicates the popularity of URL j in cluster i. Table 1 shows the

first 10 most prominent usage profiles (out of 46) and the noise profile with their descriptions. Again order or prominence is automatically determined by RFSC. This order (which is not found in FCM based algorithms) itself can give the web analyst important information about the recent trends among users of the website. This can be stated as an additional advantage of our algorithm. Also in Table 1, within each profile we show only the first 5 most popular URL's which give us an intuitive feel for the interest areas and trends that are captured in these profiles, and the cardinality of each cluster after defuzzification.

Profile #	Usage Profile	Card.	Description
1	/~eavis/comp249/ - 0.764 /~eavis/comp249/main.html - 0.632 /~eavis/comp249/assignments.html - 0.468 /~eavis/comp249/notes.html - 0.328 /~eavis/hobbit/ - 0.316	3565	Profile related to students taking comp 249 course
2	/programs/ugrad/courses.html - 0.407 /programs/ugrad/cs/cs.html - 0.215 /prospective_students.html - 0.171 /programs/ugrad/cs/curriculum.html - 0.168 /programs/ugrad/soen/soen.html - 0.134	4221	Profile related to prospective students interested in undergrad studies in computer science department
3	/~comp471/main.htm - 0.941 /~comp471/ - 0.933 /~comp471/General/ASSIGNS.htm - 0.344 /~comp471/General/NOTES.htm - 0.313 /~comp471/General/LAB.htm - 0.246	2730	Profile related to students taking comp 471 course
4	/people/people.html - 0.662 /people/faculty.html - 0.579 /current_students.shtml - 0.500 /people/graduates.html - 0.156 /people/staff.html - 0.106	4963	General profile related to faculty, students and staff of computer science department
5	/~comp239/ - 0.559 /~comp239/2005W/ - 0.552 /~comp352/ - 0.460 /~comp352/2005w/ - 0.449	1832	Profile related to students taking comp 239 and comp 352 courses together in Win 2005

	/~comp352/2005w/Info/out.html - 0.110		
6	/programs/grad/courses.html - 0.338 /programs/grad/masters/master.html - 0.222 /programs/grad/diploma/courses.html - 0.113 /programs/grad/diploma/diploma.html - 0.092 /programs/grad/diploma/comp5511.shtml - 0.081	3070	Profile related to prospective students interested in graduate studies in computer science department
7	/~comp646/winter05/labs/ - 0.638 /~comp646/winter05/ - 0.518 /current_students.shtml - 0.443 /~comp646/winter05/assignments/ - 0.409 /~comp646/ - 0.349	2798	Profile related to students taking comp 646 course
8	/~mokhov/comp346/main.shtml - 0.390 /~mokhov/comp346/ - 0.373 /~mokhov/soen321/grades.shtml - 0.346 /~mokhov/soen321/main.shtml - 0.318 /~mokhov/soen321/ - 0.315	2345	Profile related to the tutor of the comp 346 and soen 321 courses
9	/~comp218/ - 0.922 /~comp218/Comp218/Comp218WebPage/Html_Files/Main.html - 0.819 /~comp218/Comp218/Comp218WebPage/Html_Files/assignment.html - 0.319 /~comp218/Comp218/Comp218WebPage/Html_Files/SectionS.htm - 0.285 /~comp218/Comp218/Comp218WebPage/Html_Files/Slides.htm - 0.238	1711	Profile related to students taking comp 218 course
10	/~soen344/05W/home.html - 0.835 /~soen344/ - 0.781 /~soen344/05W/schedule.html - 0.662 /~soen344/05W/references.html - 0.301 /~soen344/05W/assignments/assignments.html - 0.250	1655	Profile related to students taking soen 344 course
Noise	/~ni_wang/openglspace/index.htm - 0.031 /~ni_wang/openglspace/welcome.html - 0.030 /db/db/main.html - 0.022 /~tktran/welcome.html - 0.019 /~lzhang/ - 0.018	4244	This is noise profile consisting of sessions with low membership to all the cluster. We see that it's a mix of unrelated pages with top five pages having very low popularity

Table 1 Examples of Usage Profiles discovered

3.5.4 Performance Study of ARCA and RFSC

We now compare ARCA and RFSC in terms of the similarity/dissimilarity in the resulting models and clustering times. We first present the evaluation metrics used by us for measuring the similarity between RFSC and ARCA partitions. This is followed by an analysis of the results of our experiments.

Evaluation Metrics

For similarity analysis, we have used the Rand index. We measure the similarity between clusters generated by RFSC and those generated by ARCA. Below, we briefly describe these metrics [Jain and Dubes, 1988]. Let $V = \{v_1, v_2, \ldots, v_R\}$ and $W = \{w_1, w_2, \ldots, w_C\}$ be two crisp partitions of n objects, that is, there are R clusters in partition V, and C clusters in partition W. We construct a contingency table for the two partitions. Entry n_{ij} in Table 2 is the number of objects that are in cluster v_i and also in w_j. The value $n_{i.}$ is the sum of values in row i, or the number of objects in cluster v_i, and $n_{.j}$ is the number of objects in cluster w_j.

	w_1	w_2	...	w_C	
v_1	n_{11}	n_{12}	...	n_{1C}	$n_{1.}$
v_2	n_{21}	n_{22}	...	n_{2C}	$n_{2.}$
.					.
.					.
.					.
v_R	n_{R1}	n_{R2}	...	n_{RC}	$n_{R.}$
	$n_{.1}$	$n_{.2}$		$n_{.C}$	

Table 2 Contingency Table for Partitions V and W

Considering each pair of objects x_i and x_j, there are four types of possibilities to consider, as described below.

Type 1: x_i and x_j belong to the same cluster v_a of V, and to the same cluster v_b of W.

Type 2: x_i and x_j belong to different clusters of V but to the same cluster of W.

Type 3: x_i and x_j belong to the same cluster of V but to different clusters of W.

Type 4: x_i and x_j belong to different clusters of V and to different clusters of W.

Suppose the frequencies of these four types are a, b, c, d, respectively. Then we have:

$$a + b + c + d = \frac{n(n-1)}{2}$$

These frequencies can be calculated from the contingency table as follows:

$$a = \sum_i \sum_j \binom{n_{ij}}{2} \quad ; \quad b = \sum_j \binom{n_{.j}}{2} - \sum_i \sum_j \binom{n_{ij}}{2};$$

$$c = \sum_i \binom{n_{i.}}{2} - \sum_i \sum_j \binom{n_{ij}}{2}; \quad d = \binom{n}{2} - a - b - c;$$

Using these values, we have that:

$$\text{Rand index} = (a + d) / \binom{n}{2};$$

Rand values are in the range [0, 1]. The maximum value of 1 may not be achievable when the two partitions have different number of clusters.

Similarity of partitions by Rand index

Ideally we would have liked to compare the similarity in partitioning for the whole dataset with 64,529 sessions. However, we soon found that ARCA being an iterative objective function minimization algorithm, it just takes too long for large datasets (computation times of the order of days and weeks on a single desktop computer). We then proceeded by taking random samples of the dataset of size 2000, 4000, 6000,..., in increments of 2000 and applied both algorithms. We wanted to see how far we could go

with ARCA within reasonable time frames. ARCA being a FCM based algorithm, there remains the question of specifying input parameters, number of clusters, fuzzifier m and initial partitions to be specified (as discussed earlier in section 3.2). Since our immediate goal is to measure cluster similarity, one choice was to have the same number of clusters in both ARCA and RFSC. We could then first apply RFSC and simply use the number of clusters resulting from RFSC for ARCA initialization. For each dataset we experimented with, we used the index of goodness for RFSC to find the best clustering. Thus, the number of initial clusters for ARCA was initialized to be C, the number of clusters found by RFSC. For initial partition U(0) of ARCA, we first defuzzified the RFSC clusters. Again, defuzzification causes noise sessions which have low membership with all the clusters to be separated out. We then spread the membership of each of these noise sessions equally across all the clusters as 1/C for ARCA initialization. Table 3 shows the similarity comparisons of ARCA and RFSC (ARCA time in hours, and RFSC in seconds). We see that even for a dataset of size 2000, ARCA took 6 hrs while RFSC terminates in just above 5 seconds. For a dataset of size 8000 ARCA took over 4 days. We did not experiment for ARCA with larger sizes, whereas we continued with RFSC. RFSC took around one and half hour to cluster the entire dataset of 64,529 sessions.

Rand index is a measure defined for crisp clustering. Therefore, to apply this measure, we first need to defuzzify the clusters where by we add the $(C+1)^{th}$ cluster as the noise cluster. While defuzzification separates noise points naturally for RFSC, for ARCA we set the membership cut off to 0.15 (chosen arbitrarily). That is sessions, whose memberships to all clusters are less than this value, are put into the noise cluster. We are now ready to measure the similarity of clusters obtained from RFSC and ARCA.

In Table 3, the Rand index values, which are very close to 1, indicate that the partitioning obtained by RFSC and ARCA are very close. We can also see that for different random samples of the dataset with different sizes, the Rand index values are almost the same.

Dataset size	No. Of Clusters (C)	Time		Rand index
		ARCA **(hrs)**	RFSC **(sec)**	
2000	48	6.3	5.1	0.926
4000	46	14.2	24.1	0.933
6000	47	40.8	49.0	0.928
8000	48	99.3	92.0	0.931
10000	48	-	138.7	-
12000			-	-
64529	46	-	5564.46	-

Table 3 RFSC vs. ARCA similarity comparisons

Similarity of partitions by interest areas

Though Rand index gives us an idea of the partition similarity, it would be helpful to actually understand it by looking at the underlying clusters (profiles) itself. We show clustering results for the dataset of size 2000 in Table 4. We sorted the clusters according to the cluster cardinality. The top 5 profiles for RFSC and ARCA are shown below in Table 4. Within each profile we show the first 5 most popular URL's as in Section 3.5.3.

Profile#	**Usage Profiles (RFSC)**	**Cardinality**		**Usage Profiles (ARCA)**	**Cardinality**
1.	/people/people.html /people/faculty.html /current_students.shtml /people/graduates.html /people/contacts.html	172		/programs/ugrad/courses.html /programs/ugrad/cs/comp335.shtml /programs/ugrad/cs/comp248.shtml /programs/ugrad/cs/comp353.shtml /programs/ugrad/cs/comp238.shtml	102

2.	/programs/ugrad/courses.html /programs/ugrad/soen/soen.html /programs/ugrad/cs/curriculum.html /programs/ugrad/cs/comp248.shtml /programs/ugrad/cs/comp353.shtml	163		/programs/grad/courses.html /programs/grad/masters/master.html /index.html /programs/grad/masters/comp6511.shtml /programs/grad/masters/comp691b.shtml	93
3.	/~eavis/comp249/ /~eavis/comp249/main.html /~eavis/comp249/assignments.html /~eavis/hobbit/ /~eavis/comp444/	114		/~comp471/main.htm /~comp471/ /~comp471/General/ASSIGNS.htm /~comp471/General/NOTES.htm /~comp471/General/LAB.htm	87
4.	/programs/grad/courses.html /programs/grad/masters/master.html /current_students.shtml /programs/grad/masters/comp691b.shtml /programs/grad/masters/comp6511.shtml	94		/people/people.html /people/faculty.html /people/graduates.html /people/contacts.html /~bill/	78
5.	/~comp471/main.htm /~comp471/ /~comp471/General/ASSIGNS.htm /~comp471/General/NOTES.htm /~ha_l/	88		/~eavis/comp249/ /~eavis/comp249/main.html /~eavis/comp249/assignments.html /~eavis/hobbit/ /~eavis/comp249/notes.html	75

Table 4 Usage Profiles for cluster similarity comparisons

Comparison of the profiles obtained by both techniques gives us an insight of how these techniques work. While RFSC tries to look at the relative density of data points, ARCA gives us more of an aggregate view trying to minimize the objective function and making sure that C clusters are found. Every cluster found by RFSC has its density relative to the first cluster i.e. it finds clusters in the order of decreasing significance, cluster which is obtained first is with the highest potential and the clusters that follow are in decreasing order of their potentials. RFSC clustering process is such that it tries to maximize the inter cluster distance (i.e. tries to find new interest areas); a requirement which is very

central to clustering. From Table 4 we can see that the top 5 profiles found by both algorithms are same though not in the same order of cluster cardinality.

- Profile 1 of RFSC is same as profile 4 of ARCA. This profile is related to different people (faculty, student, staff) that make up the department.
- Profile 2 of RFSC is same as profile 1 of ARCA. This profile captures the user's interest in the under grad courses offered by the department.
- Profile 3 maps to profile 5 of ARCA. This profile is related to comp249 offered in Winter 2005 and the pages for this course are under the domain of the instructor.
- Profile 4 of RFSC is same as profile 2 of ARCA. This profile indicates interest of user's in graduate courses offered by the department.
- Profile 5 of RFSC is same as profile 3 of ARCA. This profile is related to comp471 course offered in Winter 2005. We also see that one of the highly accessed pages by students taking this course is of the tutor of comp471.

We also saw that the number of profiles (interest areas) repeated in ARCA was much higher than RFSC. ARCA tries to split a dense cluster into parts so that it can meet the requirement of C clusters which may cause bias in clustering. This may also be the reason for higher cardinality of RFSC clusters as compared to their ARCA counterparts.

3.5.5 Index of Goodness

To assess how well our cluster validity index works; we experimented with several synthetically generated datasets with geometrically specified points in two-dimensions. One such dataset can be seen in Fig. 3; it contains 400 points.

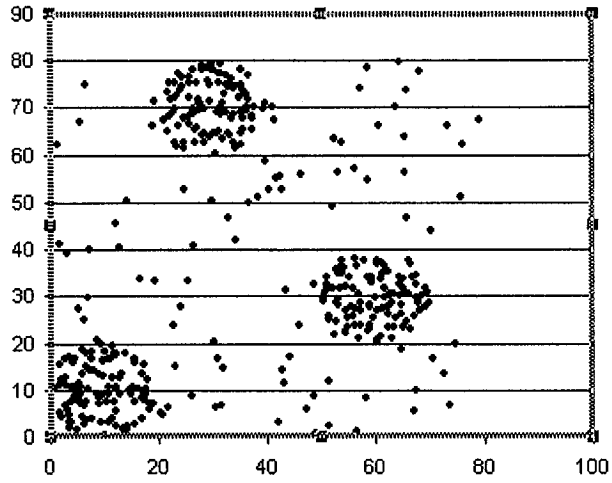


Figure 3 Synthetic dataset used for verification of index of goodness

Relational matrix R was constructed taking the Euclidean distances between these points and then normalizing them to have dissimilarity between any two points lie in [0,1]. Then we applied RFSC algorithm for values of $\bar{\in}$ and $\underline{\in}$ ranging from 0.05 to 1 in steps of 0.05, such that $\underline{\in} < \bar{\in}$ and the cluster index of goodness for each combination was calculated. For different combinations we found C to be 3, 4 or 6. Some of these values and the corresponding index is shown in Table 5.

$\bar{\in}$	$\underline{\in}$	**Index**	**C**
0.10	0.05	0.156539	6
0.20	0.10	0.041940	4
0.35	0.15	0.029624	3
0.80	0.40	0.029624	3

Table 5 Study of Index of goodness

As can be seen from this table, the best value of index occurs for C=3, which is the correct number of clusters in dataset. We also see that this can occur for more than one combination of $\bar{\in}$ and $\underline{\in}$. In any case the index for a particular value of C is always the same. Hence any of these combinations of $\bar{\in}$ and $\underline{\in}$ can be chosen for the best index

value. Choosing $\overline{\in}$=0.35 and $\underline{\in}$=0.15, we get the cluster centers at (58.13, 30.47), (30.7, 68.86), (10.7, 10.63).

3.5.6 Discussion

We make a number of important observations from this experimental analysis as summarized below:

1) ARCA being an iterative objective function minimization algorithm takes far more computation time to build the model as compared to RFSC, which largely does a single scan of the dataset for model building. This is evident from the experiment which showed that RFSC could build the model for the entire set of 64,529 sessions in just over an hour and a half, while ARCA took over four days of computation time even for the much smaller size of 8000 sessions.

2) ARCA will yield exact cluster centers, while RFSC will only yield one of the sessions as the cluster center. In our application of mining usage profiles we are not interested in finding exact cluster centers but rather profiles from the dataset.

3) ARCA is also very sensitive to user specified input parameters such as the number of clusters, fuzzifier m and the initial partition U(0). In comparison, RFSC is more stable and does not require user specified input parameters.

4) In ARCA, the partition of unity constraint on the membership values introduces an unnecessary bias for noise to also get classified into one of the C clusters. Without this constraint, noise gets handled much better in RFSC.

5) Very importantly, we see that when the number of clusters is the same, both RFSC and ARCA produce almost the same model. This we have verified using

the Rand coefficient, and also by comparing interest areas. This is a powerful result, as it does enable us to substitute ARCA with the much faster and scalable RFSC, whenever exact cluster centers are not mandatory.

Chapter 4

Recommender Models

In this chapter we discuss clustering based and association rule based recommender model. We then propose improvements over both. The first is called Fuzzy Hybrid Collaborative Filtering [Suryavanshi et al., 2005b] and the second Two Level Model-based Collaborative Filtering [Suryavanshi et al., 2005c].

4.1 Clustering Based Recommender Model

Clustering based recommender systems employ an offline clustering phase to compute a model before applying collaborative filtering [Ungar and Foster, 1998, Breese et al., 1998, O'Connor and Herlocker, 1999, Mobasher 2004]. Typically the set of sessions N_U is partitioned into C classes to extract Usage Profiles (UP) as discussed in Chapter 3. Usage profiles basically give us an aggregated view of the dataset. URLs with very low popularity in each usage profile can be pruned out.

Now for the active user session s_a for which we would like to provide recommendations, s_a is compared with different usage profiles (up_i) in order to find a profile that is most similar to s_a. It is assumed that users grouped in that cluster, which are actually aggregated in that profile, will have similar interest to that of the active user, and hence the active user may want to see those pages or items that are popular in this profile.

Let UP_t be a profile with maximum similarity to s_a. Note that there could be more than one such profile, in general, in which case we pick any one of them. For every page j in NA (set of pages not yet accessed by the active user), recommendation score is calculated as follows:

$$\text{rec\_score}(s_a, p_j) = \sqrt{up_{tj} \times sim(s_a, UP_t)} \qquad \forall j \in NA$$

where up_{tj} is the popularity of URL j (not yet accessed by the user) in profile UP_t. It follows that we could consider the top k profiles that are similar and calculate the recommendation score. If a page is recommended by more than one cluster, then the recommendation score of the page which is highest is considered. Clearly, such offline modeling improves scalability over memory-based (introduced in Section 2.4.1) and can provide on-line recommendation at highly reduced costs. But this often comes at the cost of reduced accuracy.

Our goal is to devise a CF technique whose accuracy is comparable to that of memory-based CF approach with scalability comparable to model-based (clustering) approach. We propose a technique that is a hybrid between memory-based and model-based CF. We remark that the term "hybrid" has been used in different senses by different authors. [Shahabi et al., 2001] proposed a recommender system which employs a hybrid approach that combines collaborative filtering and content-based querying to achieve better accuracy. A hybrid method based on Personality Diagnosis (PD) is proposed in [Pennock et al., 2000]. In this method, the personality type of the active user is determined and used to compute the probability of the active user requiring new items. [Nakagawa and Mobasher, 2003] also proposed a hybrid model utilizing the site structure and the degree of local hyperlink connectivity. [Burke, 2002] provides a survey on hybrid recommender

systems. Our sense of hybrid, as described, is to use both memory-based and model-based approaches for provision of more accurate recommendations at reasonable cost.

Fuzzy Hybrid CF Technique

We make use of two important properties of the model learnt from the RFSC algorithm, cluster centers and grade of memberships. RFSC computes clusters whose centers are actual sessions. If RFSC finds C clusters, then W= $\{Z_1, Z_2,\ldots, Z_C\}$ is the set of C prototypes representing these C clusters. The membership value u_{it} of each session s_t to cluster i is proportional to its distance from or dissimilarity to the cluster center Z_i. The membership values of all the sessions that are clustered are stored in the membership matrix u ($C \times N_U$). For an active session s_a, we first find the fuzzy nearest prototype [Keller et al., 1985], i.e., the cluster p to which the membership u_{pa} is maximum. We note that past like-minded user sessions of s_a will have their memberships close to u_{pa}. This simplifies the computation of k-neighbors enormously. For these k-neighbors, we compute the URL popularity for only those URLs which are in NA. If the number of desired recommendations is N, then the top-N URLs, sorted in the order of their popularity, will be presented. The above steps are incorporated into the following algorithm.

Algorithm recommend-fuzzy-hybrid-collaborative-filtering:
Input: W — set of prototypes;
u ($C \times N_U$) — membership matrix;
s_a — active session;
Output: top-N recommended URLs.
Method:
/* **STEP 1**: find a fuzzy nearest prototype */
begin
for i = 1 **to** C
Calculate D_{ia}, the dissimilarity between s_a and the i^{th} cluster prototype Z_i;

endFor
find prototype Z_p such that D_{pa} is minimum;
compute membership of s_a to p^{th} cluster using

$$u_{pa} = e^{-\alpha D_{pa}^2}$$

endBegin
/* **STEP 2**: find k-nearest neighbors depending on membership */
begin
Set k, a value in $\{1,\ldots,N_U\}$;
Initialize neighbors = 0;
for i = 1 **to** N_U
if (neighbours $\leq$ k) **then**
Include s_i in the set of k-nearest neighbors;
Increment neighbors by 1;
else
Let s_m be the farthest of the k-nearest neighbors
if ($| u_{pi} - u_{pa} | < | u_{pm} - u_{pa} |$) **then**
Delete s_m from the set of k-nearest neighbors;
Include s_i in the set of k-nearest neighbors;
endIf
endIf
endFor
endBegin
/* **STEP 3**: find the url popularity */
begin
for All url j in NA
Initialize $up_j = 0$;
for All sessions s_k in the set of k-nearest neighbors
weight (s_k, s_a) = similarity between s_a and s_k;
$up_j = up_j + s_{kj} *$ weight (s_k, s_a);
endFor
endFor
endBegin
return top-N most popular URLs;
endAlgorithm

Fuzzy K- nearest prototype

In step 1 of the above algorithm, instead of finding a fuzzy nearest prototype, we could find fuzzy K-nearest prototypes [Keller et al., 1985]. Steps 2 and 3 can be carried out for each cluster in the same way, and at the end the URL popularity scores from each cluster can be combined. Also if a URL is recommended from more than one cluster, then we

consider maximum popularity score from all the contributing clusters. Results from our experiments show that this approach leads to increased accuracy of recommendation, but with slight increase in computation time.

Binning

In step 2 above of finding k-nearest neighbors, we use equal-depth binning technique [Han and Kamber, 2000] to speed up the search of k-neighbors. Our algorithm uses the membership matrix u, which contains the memberships of each session to different clusters. Each row can be sorted in descending order of the membership values. The N_U interval is then divided into b bins, where each bin contains N_U/b elements. We build a bin index for each row, which has b elements each of which contains a value of membership at the beginning of each bin. Fig. 4 illustrates an example of this bin index in which we have 16 sessions and 3 clusters, i.,e., $N_U = 16$ and C=3. The membership matrix u is sorted in descending order of the membership grades of the sessions. We have divided this N_U interval into 4 bins, i.e., b=4, and the number of sessions in each bin is 4. A bin index is a (v, r) pair, where v is the value of the first element in each bin and r is a pointer to that bin. A bin index is constructed for every cluster, and hence the bin index is essentially a (C × b) matrix. So for the fuzzy nearest prototype p, we locate the bin which has elements or sessions whose membership is close to u_{pa}. Now within this bin, we find the k-nearest neighbors with similar memberships. The number b of bins should be selected optimally depending upon N_U. If b is large, then the number of sessions in each bin will be small but the bin index can be huge. On the other hand if b is small, then although searching through the bin index can be faster, searching within each bin may

take longer. Depending on b and the number of like-minded users being searched, the bins around may also need to be probed.

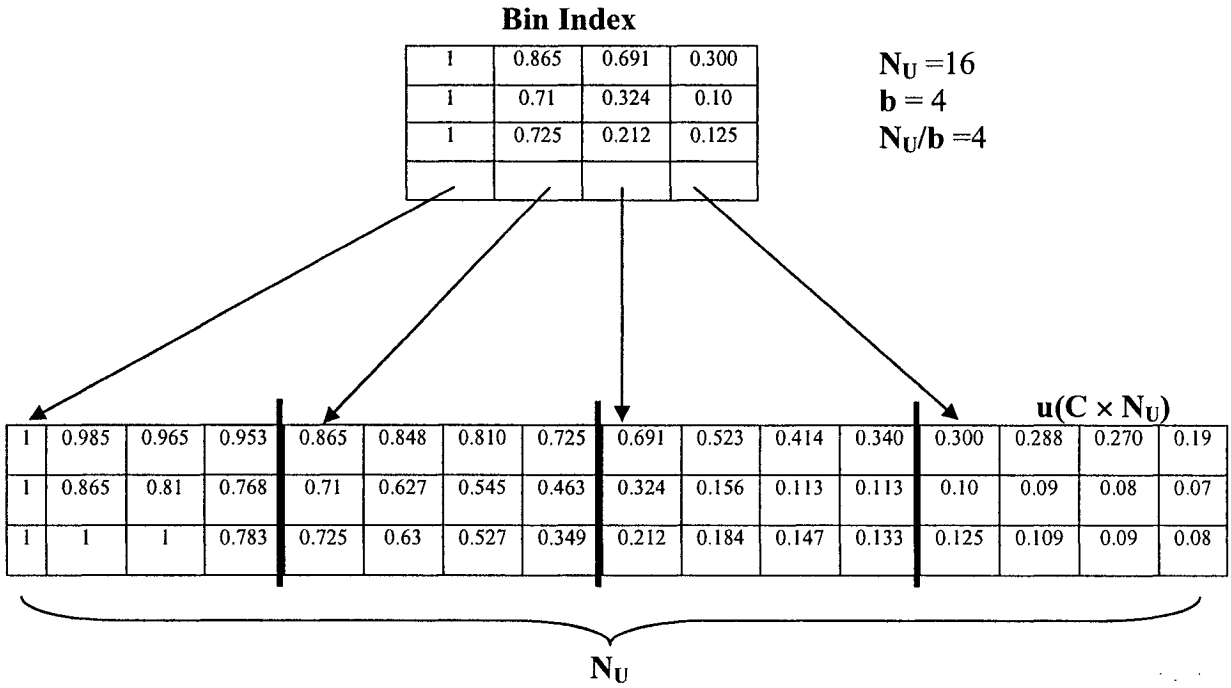


Figure 4 Example for Illustration of Binning

4.2 Association Rule Based Recommender Model

Association rule mining has also been used in order to produce a model for recommendation or personalization systems. As introduced in Section 2.3.1, an association rule is a statement of the form X → Y, where X and Y are sets (of items, or web pages in our context). Rule X→Y indicates that whenever a session s in the dataset D contains itemset X, then s also contains itemset Y with some certainty, called as confidence of the rule. In a typical association rule based recommender system [Sarwar et al., 2000b], first, all association rules are discovered from historic preference (explicit or

implicit) database D. Then preferences of the active user s_a are matched against the items in the antecedent (X) of each rule. The items on the right hand side of the matching rules are sorted according to the confidence values, and the top N ranked items are recommended. As web usage data is highly sparse, there may be a large number of short rules due to which there is often no exact match between the preferences of the active user and antecedent of the rules. To improve the performance of recommender systems, [Sarwar et al. 2000a] consider dimensionality reduction technique. [Fu et al., 2000] suggest an alternative by finding rules with best match. The degree of intersection of rules with active session can be used as a weight to determine the recommendation set. [Mobasher et al., 2001b] used a sliding window w over the current active session. The size of this window is iteratively decreased until an exact match with the antecedent of a rule is found. For this, the maximum size of the window has to be chosen, which is often chosen arbitrarily as there is no prior knowledge for this. Another problem with this technique is that it requires repeated search through the rules in case no match is found.

Also, during the learning phase, a problem with using a global minimum support threshold in association rule mining is that the discovered patterns will not include "rare" but important items which may not occur frequently in the user purchase transaction or access sessions data. This is particularly important when dealing with web usage data; it is often the case that references to deeper content or product-oriented pages occur far less frequently that those of top level navigation-oriented pages. In addition, during the online recommendation phase, many rules may not even be relevant to the active user. What is required is to search through only those rules that are relevant and meaningful to the

active user and select the rules that best (may not be exact) match the preferences of the active user.

Two Level Model Based CF Technique

With this background, we now present the two level model-based CF technique. In the first level, RFSC algorithm is used to cluster the access sessions. Let $W= \{Z_1,Z_2,\ldots,Z_C\}$ be the set of C prototypes. Next, we defuzzify the clusters by assigning the sessions to a cluster to which its membership is highest. Defuzzification causes noise to be separated from the real structure in the data as shown in section 3.5. In the second level, we find association rules within each cluster. In our work, we restrict this process to generation of association rules with a single consequent. In general, this method is applicable for association rules with more than one consequent as well. For an active session s_a, our algorithm first finds the nearest prototype, then matches the s_a with the antecedent of each rule in that cluster to find the match score for each rule. This is weighed with the confidence of each rule to obtain the recommendation score (rec_score) of the page which is the consequent of that rule. If the number of desired recommendations is N, then the top-N URLs, sorted in the order of their popularity, will be presented. This segregation of interest areas and then mining association rules within each area provides more pertinent recommendations as shown by our experimental results. Algorithmic details are as follows.

Algorithm recommend-two-level-model-based-collaborative-filtering:
Input: W, set of prototypes;
Association rules for every cluster;
s_a, active session;
Output: top-N recommended URLs.
Method:
/* **STEP 1**: find the nearest prototype */
begin

```
        for i = 1 to C
            Calculate D_ia, the dissimilarity between s_a and the i^th cluster prototype Z_i;
        endFor
        find prototype Z_p such that D_pa is minimum;
      endBegin
/* STEP 2: find recommendation score for every association rule in cluster p */
    begin
        for i = 1 to NumberOfRules in p
            match(s_a, r_pi) = cosine (s_a, r_pi);
            rec_score(s_a, r_pi) = match(s_a, r_pi) * confidence (r_pi);
        endFor
    endBegin
/* STEP 3: find the url popularity */
    begin
        Prune every rule whose consequent is not in NA;
        If multiple rules recommend the same consequent, choose the rule with highest
        recommendation score;
        Sort the rules according to their rec_score;
    endBegin
    return top-N most popular URLs;
endAlgorithm
```

Furthermore, in step 1, instead of finding the nearest prototype, we can find K-nearest prototypes. Steps 2 and 3 are then carried out for each cluster, and at the end, the recommended URLs from each cluster can be combined.

4.3 Experiments and Results

In this section, we first present the evaluation metrics we have used for measuring recommendation quality, and then proceed with a description of our experiments and the resulting values for these metrics.

4.3.1 Evaluation Metrics

In a recommendation system, a possible measure for efficiency is the time taken by the system for producing an online recommendation, and for effectiveness one could use prediction quality. Given any dataset, we first divide it into two parts: the training set and

the test set. For every session in the test set, we hide some pages in this session, called the Hidden set. Our algorithm then works on the training set and, for every session in the test set, generates a set of recommendations. Let top-N denote the set of TOPN number of pages recommended.

Recall and precision are two quantitative measures which have been widely used for effectiveness [Sarwar et al., 2000b; Breese et al., 1998; Pennock et al., 2000]. Recall is a global measure that corresponds to the proportion of relevant recommendations that have been retrieved by the system, i.e., the proportion of items in the hidden set that are correctly recommended. The value of recall tends to increase as TOPN increases.

$$\text{Recall} = | \text{Hidden} \cap \text{top-N} | / | \text{Hidden} |$$

Precision measures the average quality of an individual recommendation. As TOPN increases, the precision of each recommendation decreases.

$$\text{Precision} = | \text{Hidden} \cap \text{top-N} | / | \text{top-N} |$$

A measure F1 which combines recall and precision with equal weights has also been suggested and is defined as follows:

$$F1 = (2 \times \text{recall} \times \text{precision}) / (\text{recall} + \text{precision}).$$

4.3.2 Performance Study of CF Techniques

In our different experiments reported in this section, we used the same dataset described earlier in Section 3.5.1. We divided (randomly) the 64,529 sessions in this dataset into two:

(1) the training set, with 51624 sessions (80%) and

(2) the test set, with 12905 sessions (20%).

The average length of a session in the training set was 6.99 and sparsity level was 0.999292, whereas for the test set, the average session length was found to be 7.02 and sparsity level was 0.998954.

We compare the predictive ability of five different approaches:

1. Memory-based (k- nearest neighbor)
2. Pure clustering (RFSC) based recommendation model (section 4.1)
3. Fuzzy hybrid CF technique
4. Pure association rule based recommendation model (section 4.2)
5. Two level model-based technique.

For experimentation, [Breese et al. 1998] proposed two protocols. In the first protocol, called *All-but-1* protocol, one of the pages is randomly selected and withheld from each session in the test set. The CF algorithm is then used to predict this page. In the second protocol, called *Given t*, they suggest selection of some t number of pages from each session in the test set and use CF algorithm to predict the rest of the pages. In our experiments, we used a protocol similar to these, described as follows. We randomly selected approximately 80% of pages in each session in the test set as seen and tried to predict the other 20% of pages which were hidden using the above algorithms. Thus, if a test session is of length 3, we withheld one page, and if the session is of length 100, we withheld 20 pages to measure the predictive ability of the algorithms used.

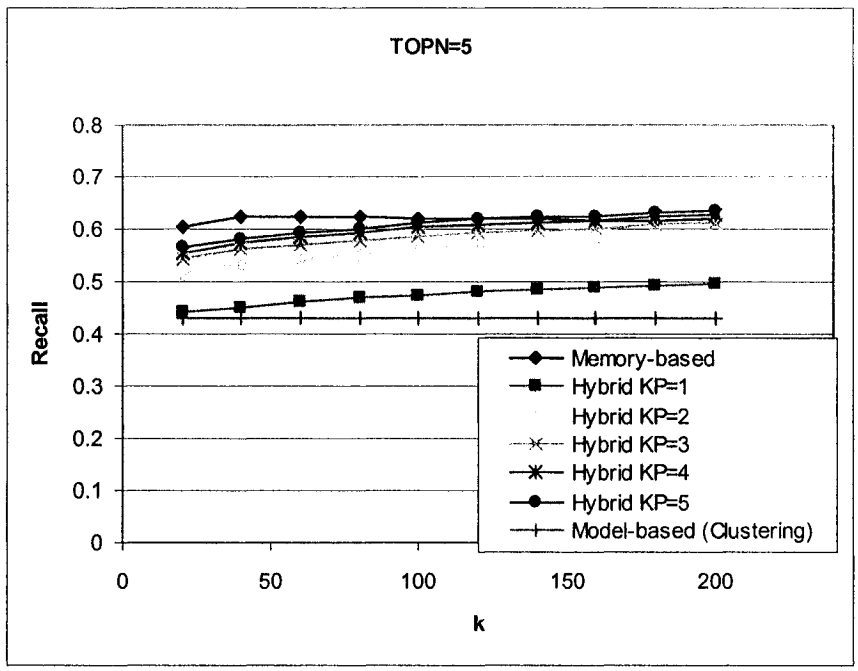
We applied RFSC to the training set which consists of 51,624 sessions, and used the goodness index to obtain a good clustering. In all, 47 clusters were found, (C=47). The total time taken for this modeling process was slightly above an hour. We found 3366 sessions having low memberships with all the clusters i.e. noise sessions. Fuzzy hybrid

CF makes use of the membership matrix u ($C \times N_U$), having membership grade of each session to C clusters, as well as the set of prototypes $W = \{Z_1, \ldots, Z_C\}$. For two level model-based CF, we defuzzified the clusters. Now association rules are discovered within each cluster. We also applied association rule mining to the entire training dataset for pure association rule based model. A critical factor here is to determine the global support level for mining association rules. Increasing the support level decreases the number of rules, and hence we may miss some important rules. On the other hand, decreasing the support level may lead to identifying a large number of irrelevant rules. In our experiments, we used different support values. In the results shown here we have used a support value of 0.01 for pure association rule based model, and used 0.1 for mining association rules within each RFSC cluster for two level model-based CF.

Effectiveness Comparisons

We will use the following notation in the explanation of experiments and results. We use k for the number of nearest neighbors being considered, KNP for the number of nearest prototypes, and TOPN for the number of top N popular pages for recommendation. Fig. 5 and Fig. 6 show the comparison of recommendation quality in terms of recall and precision (F1 can be inferred from these two) of memory-based, model-based (RFSC clustering), and the hybrid (Fuzzy 1-Nearest Prototype (1-NP), 2-Nearest Prototype (2-NP), 3-Nearest Prototype (3-NP), 4-Nearest Prototype (4-NP), 5-Nearest Prototype (5-NP)). Fig. 5 shows recall and precision values for TOPN=5, while Fig. 6 shows these values for TOPN=10. This choice of TOPN should be made reasonably and not arbitrarily. Obviously, lower values of TOPN will lead to lower accuracy whereas higher values will increase the accuracy of the results. For personalized recommendations, it is

desired to show the users a small number of specialized products, items, or links rather than providing overwhelming choices. Presenting a large number of links or items would in fact work against the initial goal of personalization itself. As can be seen in Fig. 5 and Fig. 6, memory-based approach seems to achieve high recall and precision for around k=100, after which they drop as we consider more number of neighbors. The recall and precision measures for cluster based recommender model are shown as lines parallel to the x–axis, since there is no notion of k in the model-based approach. With fuzzy hybrid CF, the quality keeps increasing as more neighbors are considered. While the results of 1-NP seem to be better than those of pure model-based, considerable increase in quality is observed as we consider more nearest prototypes (2-NP, 3-NP, 4-NP, 5-NP in that order). In fact for both cases, we see that fuzzy hybrid CF almost reaches or is even better than memory-based approach.



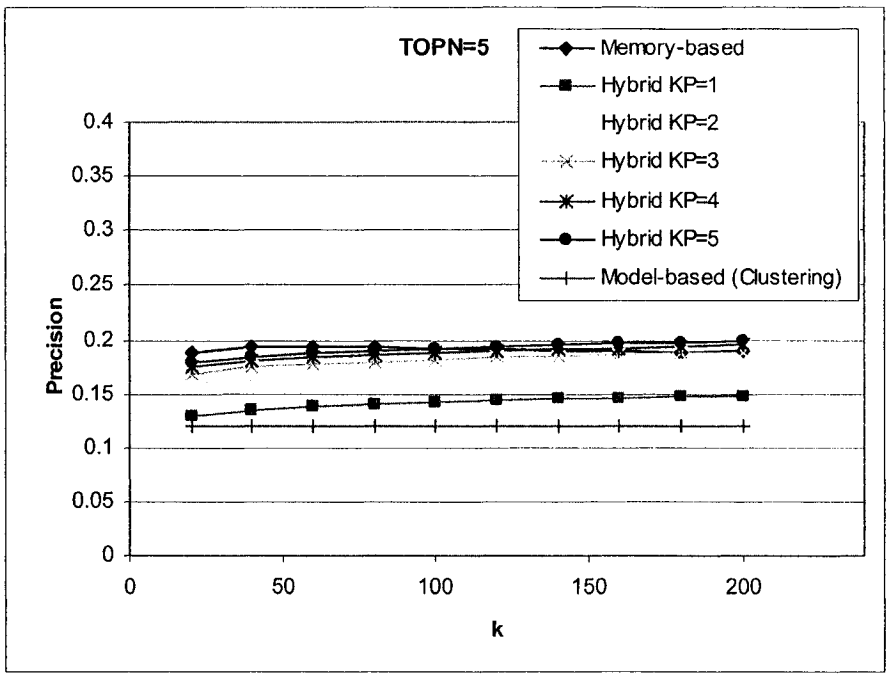
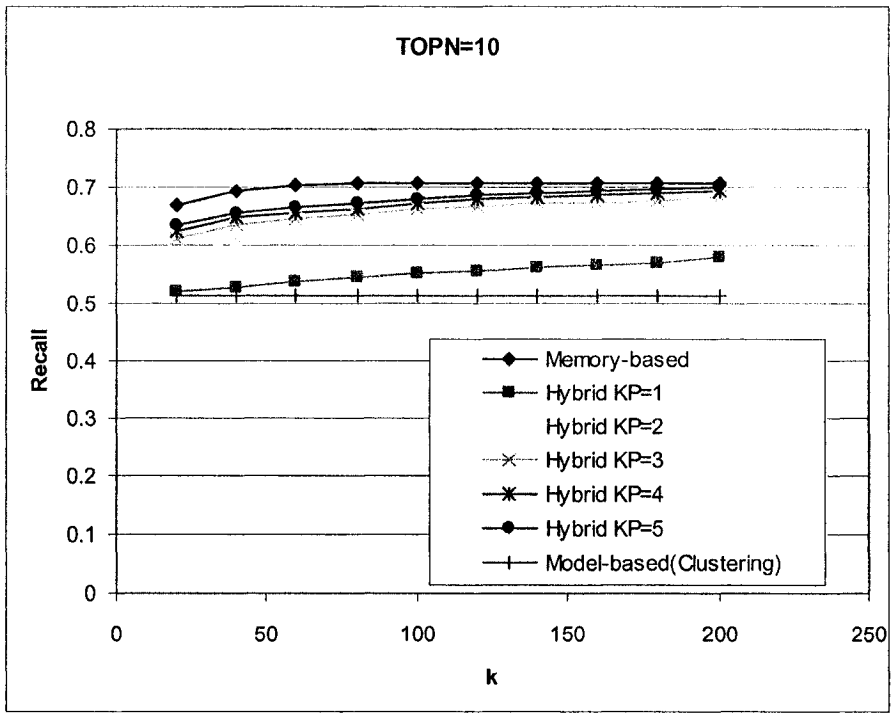


Figure 5 Recommendation effectiveness, TOPN=5



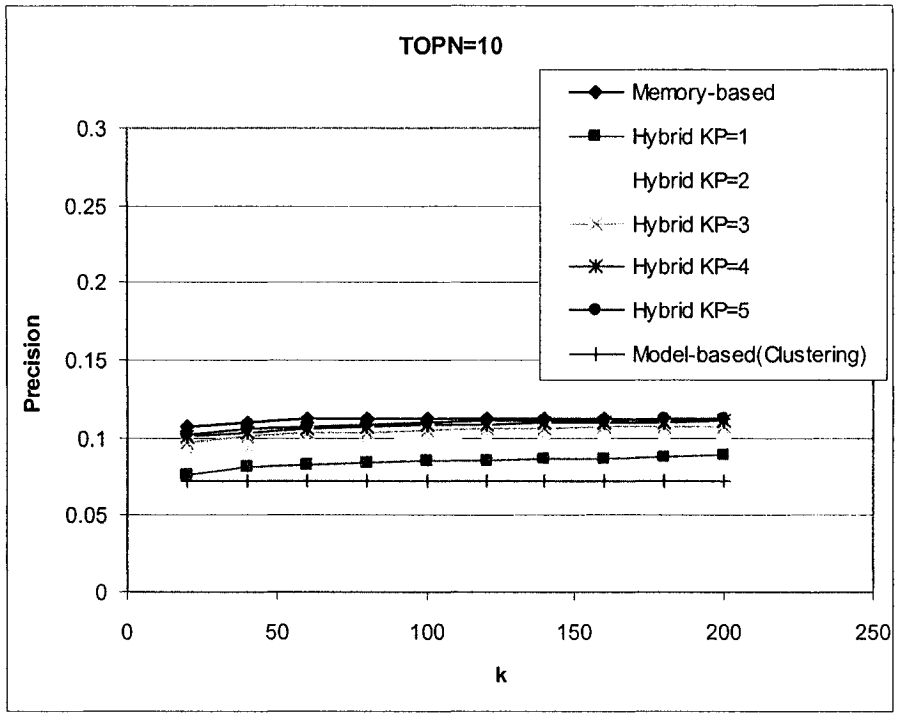


Figure 6 Recommendation effectiveness, TOPN=10

The goal in two level model based CF is to improve the effectiveness of association rule based recommender systems for sparse web usage data. Fig. 7 shows the comparison of recommendation quality of memory-based, model-based (pure association rule model), and two level model-based CF (1-NP, 2-NP, 3-NP, 4-NP, 5-NP). We show these results for TOPN =5, 10, 15, 20. As TOPN increases, recall increases, but precision decreases. There is no notion of k neighbors here. We see that considerable increase in effectiveness is achieved by two level model-based CF over pure association rule model. In fact for lower values of TOPN, the effectiveness achieved is comparable to memory-based. This shows how segregation of interest areas and then mining association rules within each area can provide more pertinent recommendations.

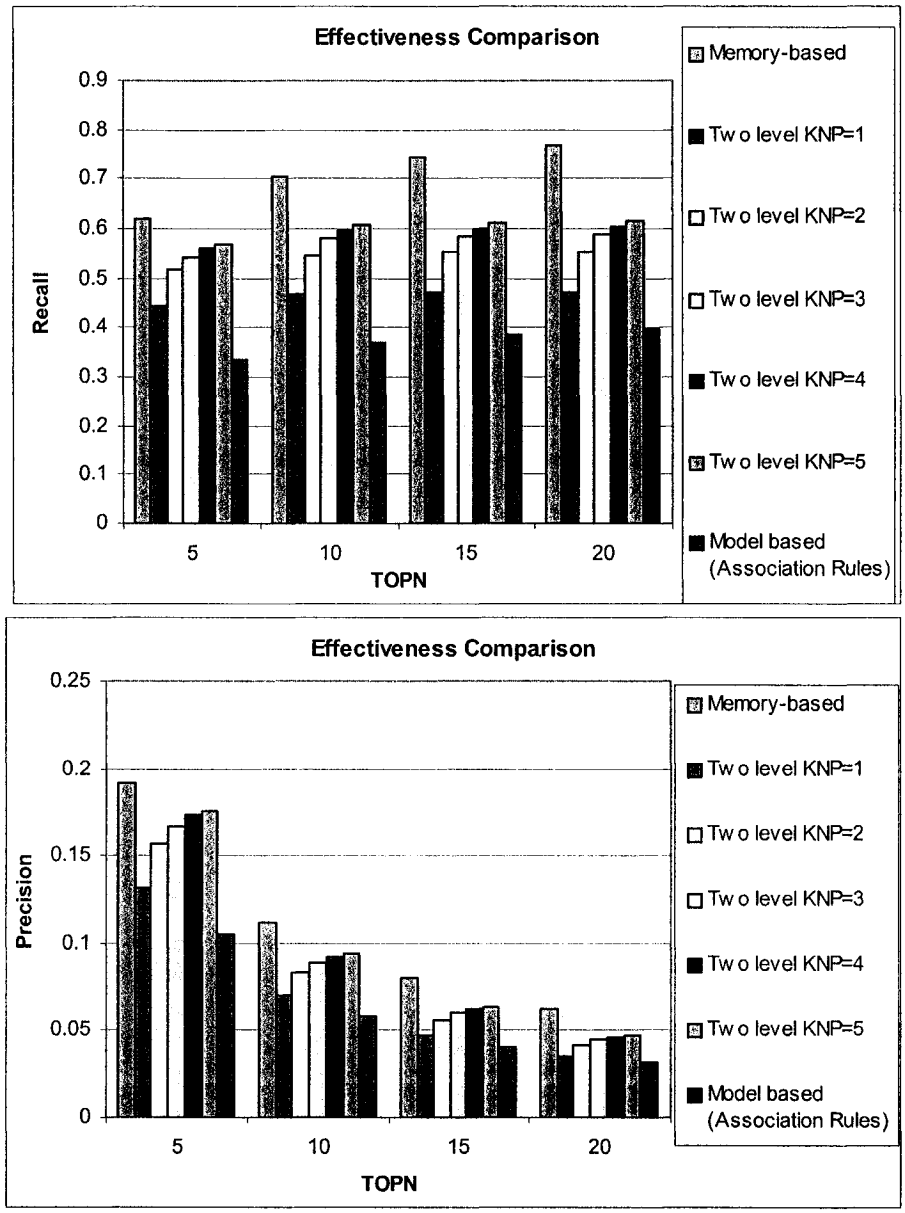


Figure 7 Improvements achieved with two level model-based CF

Below, we provide examples of some rules which were found with a high confidence (above 95%) in different clusters, but were missed when association rule mining technique was applied on the dataset as a whole.

[/~comp676/], [/~comp676/Winter05/COMP6761_ASSIGNS.htm],
[/~comp676/Winter05/index.html] → [/~r_rajago/graphics/COMP471_Tutorials.htm]

[/~nacemian/WebPage/HTM-HTML/Main.html], [/~p_chua/rmp/p.htm], [/~nacemian/encs5821.html]→ [/~p_chua/Destination/MainDestination.htm]

[/CONCEPT/index.html], [/db/db/index.html] → [/research/research.html]

After investigating these patterns we came up with these possible explanations:

- Rule 1: COMP 6761 is an advanced course in computer graphics which was offered in Winter 2005. Students have been visiting tutorials of the prerequisite of this course when solving problems in assignments of COMP 6761.
- Rule 2: This is another interesting rule. ENCS 5821 course was taught by a visiting professor in winter 2005. Students were interested in knowing more about this visiting professor.
- Rule 3: This rule indicates that users who are interested in research at CSE department have shown interest in both CONCEPT and database research groups.

Figure 8 shows comparisons of all the five techniques for different TOPN.

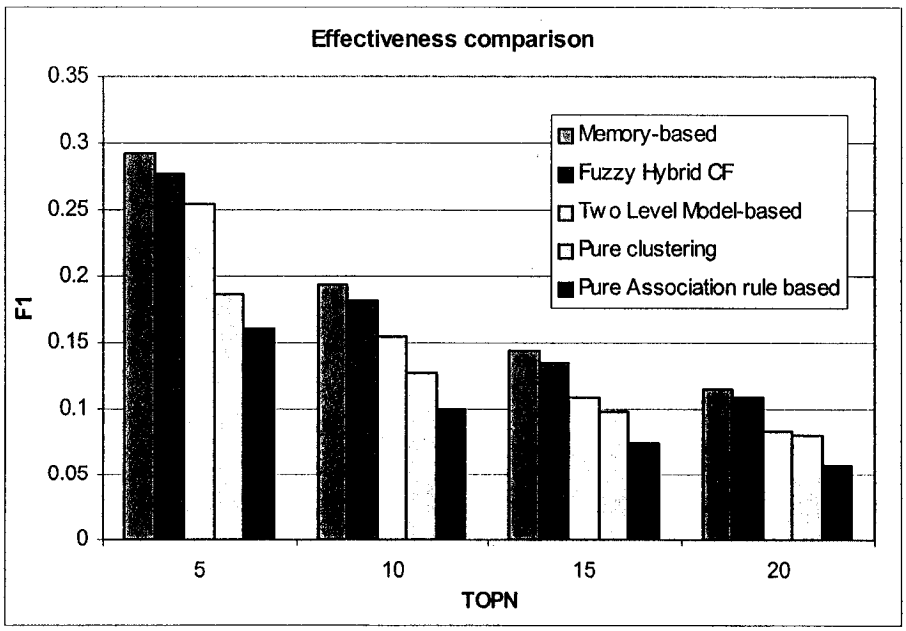


Figure 8 Quality comparison of five CF techniques

We use the F1 measure, and fixed k to 100 and KNP to 3. F1 being the combined measure, we can see that the net effect of increase in TOPN is decrease in F1. We see that fuzzy hybrid CF and two level model based CF achieve considerable improvement over their pure model-based counterparts, while the quality of fuzzy hybrid CF is as good as memory-based CF.

In our fuzzy hybrid CF, we make use of the membership matrix $u(C \times N_U)$, which stores the membership grades of different sessions with respect to different clusters. Instead of storing the memberships, we now store dissimilarities of sessions with respect to the cluster centers. Fig. 9 illustrates the results of this comparison (k=100, TOPN=5).

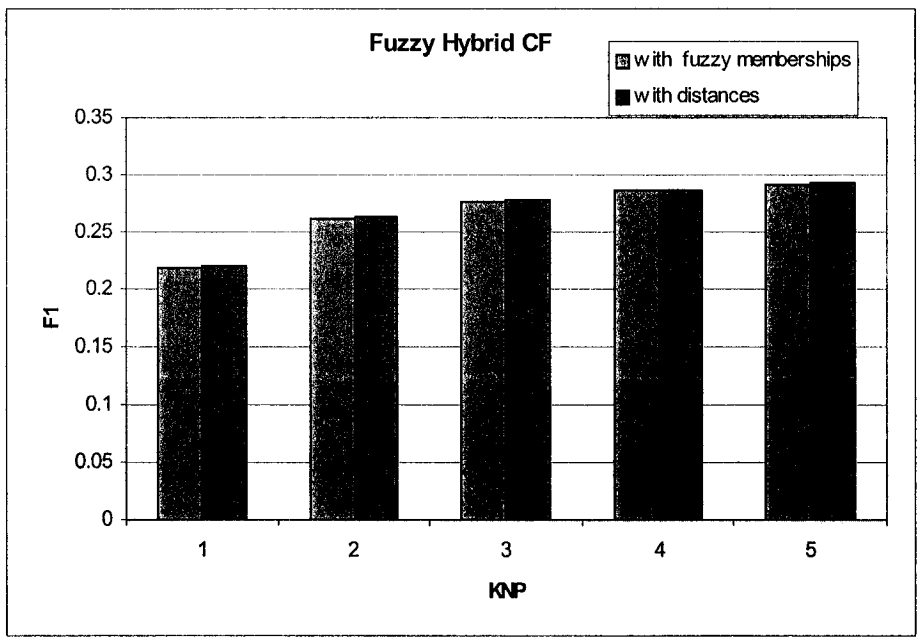


Figure 9 Comparison of Effectiveness (membership vs. distances)

We expected that using the dissimilarity measures instead of membership would give use effectiveness equal or better than using memberships (as memory based approach relies purely on distances). Experimentation with different random samples of the dataset

showed almost the same results, that is, membership as well as distances achieve the same effectiveness.

Efficiency Comparisons

While the aforementioned mentioned techniques achieve higher accuracy, we need to convince ourselves that this improvement in accuracy comes at reasonable computational cost and time so that we can make online recommendations on the fly. Fig. 10 compares the online recommendation time/user on a standard desktop system for memory-based and fuzzy hybrid CF (1-NP, 2-NP, 3-NP, 4-NP, 5-NP).

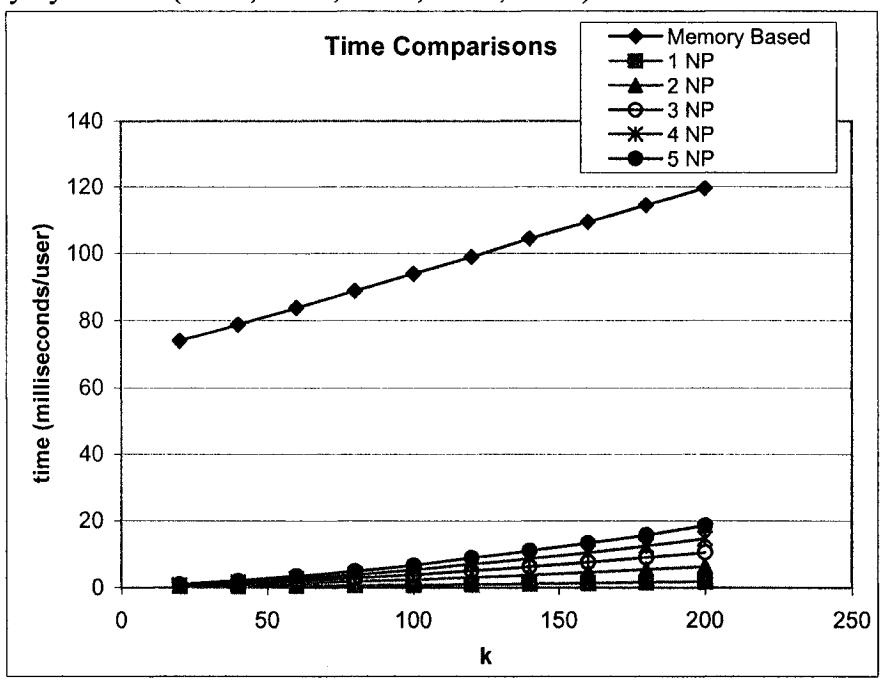


Figure 10 Comparison of efficiency in terms of online recommendation time

The recommendation time for memory-based approach is much higher than our hybrid approach. On average, time taken for memory-based approach is 30 to 40 times more than the time taken by the hybrid approach (3-NP considering fair comparison). This will continue to increase linearly as more and more sessions are added. Also for 1-NP, 2-NP, 3-NP, 4-NP, and 5-NP, the time increases in that order, but at the same time, this small

increase in time results in considerable increase in recommendation quality as seen above in effectiveness comparisons. The average time taken by model-based approach (RFSC) measured was about 0.187679 milliseconds/user. The time taken by the hybrid approach is just about 5 to 7 times more on the average than the model-based approach.

For two level model-based approach, the online recommendation time is considerably higher, at around 40 milliseconds/user, while it is around 10 milliseconds/user for pure association rule based recommender model. This could be further improved if we store the rules in a way to speed-up the search. One such technique is proposed in [Mobasher 2004]. Fig. 11 shows the time comparisons for all the five techniques for different KNP. We have fixed k=100 for this comparison.

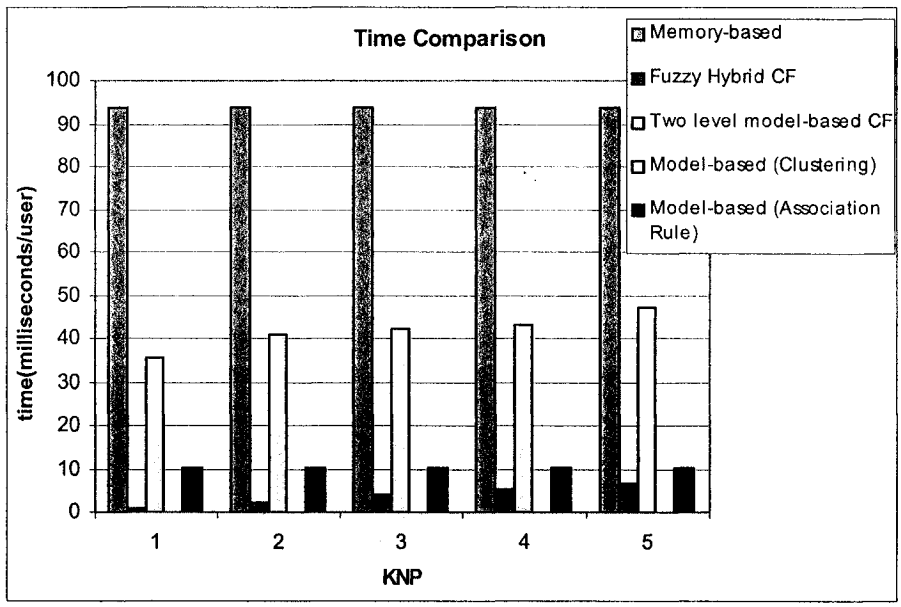


Figure 11 Comparison of efficiency for five CF techniques

We observe here that time taken for the improvements namely fuzzy hybrid CF and two level model-based CF are relatively higher than their pure model-based counterparts. Pure cluster based recommender model requires the least recommendation time (so small

that it is almost invisible in the figure) while memory-based approach requires the highest.

Binning

We use equal-depth binning technique to further enhance the efficiency of our fuzzy hybrid CF. Resulting improvements are shown in Fig. 12. The size of the training set is N_U=51,624. We choose the number of bins as b=500, i.e. except for the last bin which gets 227 sessions, all other bins include103 sessions. This is not an arbitrary choice but operationally motivated, explained as follows. In our experiments the maximum number of neighbors (k) is 200. We always search in the bin in which sessions with membership closest to the membership of s_a would be found, as well as the bin to its left and the bin to its right. If the bin in which sessions with membership closest to the membership of s_a is the first bin then we search in that bin as well as the bin to its right and if it is the last bin then we search also in the bin to its left. Hence the choice of b. We fixed k to 100 and TOPN to5.

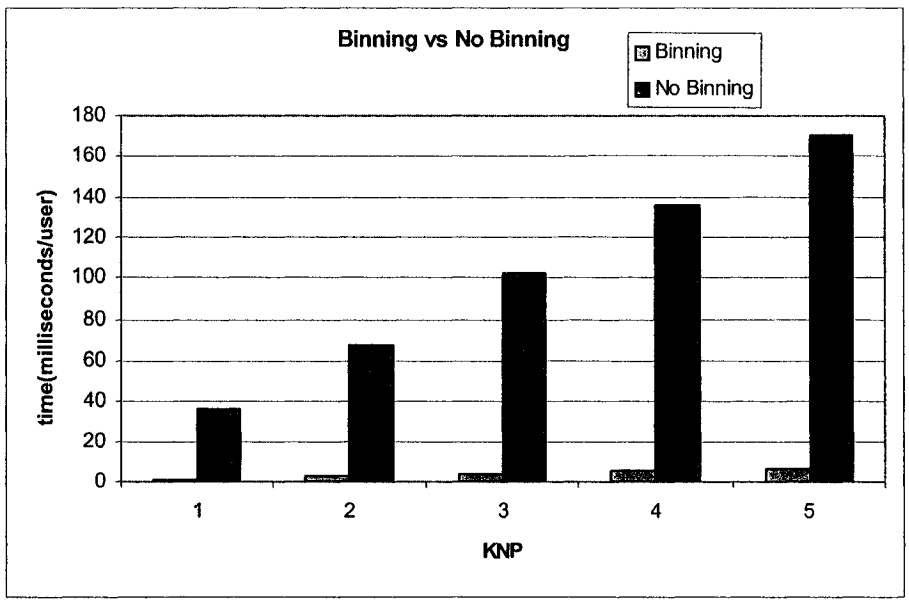


Figure 12 Comparison of efficiency (Binning)

Recommendation time/user is shown for different values of nearest prototype (KNP). Binning gives us a performance increase by a factor of 25-35 on an average. Also binning did not affect the recommendation effectiveness negatively. In fact, even a quality raise (marginally) can be seen in some cases.

4.3.3 Discussion

Our comparative study of CF techniques yielded the following interesting observations:

1. Not unexpectedly, memory-based CF technique yields accurate recommendations but at rather high computation costs. In comparison model-based CF techniques do indeed take much less time with some loss of accuracy. On the average, reduction of computation time by a factor of 40 can be achieved, making them highly suitable for real time recommendation.

2. Improvements to model-based techniques do result in substantial increase in recommendation efficiency, with modest increases in computation times. Use of learned access behavior characteristics improve both cluster based and also association rule based recommendations. Of these, the fuzzy hybrid CF, which performs k-nearest neighbor using membership values, yields the best results, almost the same as the memory-based.

3. In fuzzy hybrid CF efficiency can be further improved by employing equal depth binning. Binning gives us a performance increase by a factor of 25-35 on an average. This is one of the major factors in being able to achieve speed comparable to pure model-based techniques.

4. We also experimented by considering distances of sessions with respect to centers rather than fuzzy memberships. We observed that the quality for both memberships as well as distance is almost the same and using distance or dissimilarity measures does not provide any performance advantages.

To the best of our knowledge, this kind of elaborate experimental study is the first of its kind and we believe that our overall experimental framework, experimental results and observations are of interest to the web personalization research community and also to other disciplines concerned with fuzzy techniques for modeling of large datasets.

Chapter 5

Incremental Relational Fuzzy Subtractive Clustering

In the last two chapters, we explored how web usage models and profiles, which capture significant interests and trends among users, can be extracted from usage data and used to improve the user's experience through personalized recommendations. However, a popular website is visited frequently by a large number of users with a variety of needs. The browsing behavior of users of such sites is not fixed or static but changes dynamically over time. If these profiles or models are fixed and do not adapt to new interests and usage, this may lead to degradation of the system over time. Irrelevant and mis-targeted recommendations annoy the users leading to low impact or ineffective personalization systems. Most existing techniques for modeling web usage are static or inefficient, mainly due to the fact that the time complexity to compile the data into a model is in general prohibitive and adding one new data point may require a full recompilation of the model [Pennock et al., 2000]. If personalized recommendations have to succeed, it is vital to develop maintenance techniques, which can adapt to changes in the usage model in non-stationary web environments. We call such profiling that

dynamically adapts to new usage data without complete remodeling as *Adaptive Web Usage Profiling* [Suryavanshi et al., 2005d][4]. This forms the main focus of this chapter.

5.1 Overview of Cluster Maintenance

A clustering algorithm should have the following desirable properties [Rijsbergen, 1979, Can et al., 1987]:

1) Stability -- produce a clustering which is unlikely to be altered drastically when new objects are added.

2) Robustness -- small errors in the description of the objects may only lead to small changes in the clustering.

3) Order insensitivity -- composition of clusters is independent of the order in which objects are processed.

4) Maintainability -- handle growth efficiently, i.e., maintenance of clusters should be practical and efficient.

Most existing clustering algorithms are not suitable for maintaining clusters in a dynamic environment, and often the problem of updating clusters is not solved without a complete reclustering [Rijsbergen, 1979, Can et al., 1987, Charikar et al., 2004, Tasoulis et al., 2005]. One reason is that clustering and maintenance techniques are merely based on heuristics which very often make reclustering of the entire dataset more practical. Also most clustering algorithms suffer from limitations such as requiring the specification of correct number of clusters, sensitivity to initialization of clusters, and to noise.

[4] An extended version of this paper has been submitted as a chapter for WebKDD book edited by Nasraoui, Zaiane, Spiliopoulou, Mobasher, Masand, and Yu.

One of the early works done in dynamic cluster maintenance in information retrieval is [Can et al., 1987], which proposed a cluster maintenance scheme based on the notion of cover coefficient. An incremental document clustering algorithm is proposed in [Charikar et al., 2004], which attempts to maintain small diameter clusters as new points are inserted in the dataset. The authors also studied the dual clustering problem, where the clusters are of fixed diameter, and the goal is to minimize the number of clusters. [Ester et al., 1998] proposed an extension of GDBSCAN algorithm for large updates in a data warehouse environment. [Tasoulis et al., 2005] introduced an extension of the k-windows algorithm that can discover clustering rules from a dataset in a dynamic environment. In web usage mining, [Shahabi et al., 2002] studied dynamic clustering as an approach to make the cluster model adaptive to short-term changes in users' behaviors. They argue that web usage mining systems should compromise between scalability and accuracy to be applicable to web sites with numerous visitors. They show that while accuracy of dynamic clustering is low, it helps achieve adaptability. [Nasraoui et al., 2003] were among the first to propose a framework of mining evolving user profiles through a new scalable clustering methodology, inspired from the natural immune system, which is capable of continuously learning and adapting to new incoming patterns.

Our work differs from earlier maintenance schemes in several ways, most notably the following:

- Many of the aforementioned maintenance schemes are for crisp clustering. Our maintenance scheme is for RFSC, which is a fuzzy clustering algorithm. There has been little work reported is this area.

- We strongly believe that computation of similarity between the clustering obtained after application of any maintenance algorithm and complete reclustering is necessary to validate a maintenance scheme. Unlike in many of the above, in this work we show the validity of using incremental RFSC by computing similarity.

- To the web analyst, it is often unknown as to when a complete remodeling is required. It is also useful to know when the user access patterns have drastically changed. This would help the analyst to set historic time windows when remodeling operation is applied. For this we provide a quantitative definition of an "impact factor." Thresholding of this factor can indicate as to when complete remodeling becomes necessary.

5.2 Maintenance Scheme for Relational Fuzzy Subtractive Clustering

Though the clusters obtained from usage data manifest the interests and trends among the users accessing the site at the time the clustering operation was applied, the interests and needs of users change dynamically over time. New logs are continuously created as users access the website. We thus require a cluster maintenance scheme by which these changing trends and patterns can be captured without having to frequently apply this relatively expensive operation of reclustering of large volume of old and new data together.

Cluster maintenance is not a complete alternative to reclustering. By its very definition, cluster maintenance tries to incorporate newly arrived data into the existing model, while maintaining the profiles created earlier from the original set of data. This at best can result in a close approximation to clustering of the complete data, including both old and new. Clearly, reclustering will always give more accurate results and is required to be done periodically. Cluster maintenance however enables us to continuously adapt to the dynamic and changing environment in a much less expensive manner in terms of computation times and resources, and which also allows subsequent maintenance even after reclustering. Thus a "balanced" combination of full data clustering and cluster maintenance is ideally suited for dynamic environments. In this section, we introduce an extension of the RFSC algorithm, which we refer as the *incremental RFSC* algorithm, used as the main component for our usage profile maintenance scheme. We will also introduce a measure that helps decide the stage at which a complete reclustering is required.

Let us recall the results from an initial RFSC clustering:

- We have a dataset of N_U sessions or objects and have C clusters.

- $u(C \times N_U)$ denotes the membership matrix, which contains the membership values of each object to each of the C clusters.

As new session data is received, these are updated using incremental RFSC until reclustering of the complete dataset is required. This sequence of incremental RFSC and reclustering constitutes the maintenance scheme.

Let W= $\{Z_1, Z_2,\ldots, Z_C\}$ be the set of C prototypes representing the C clusters. We also store the corresponding potentials using which these C prototypes were chosen as the cluster centers, P= $\{P_1, P_2,\ldots, P_C\}$. These potentials have a natural ordering (i.e., descending) as RFSC finds the cluster with the highest potential first, followed by the next highest potential, and so on.

Let x_{new} be a new object added to the dataset. Depending on x_{new}, we have three possible situations:

Case i) x_{new} has a high membership value with respect to one of the existing C clusters, and hence its potential is such that it becomes a core member of that cluster. Or x_{new} has a very low potential in which case it is probably a noise.

Case ii) The potential of x_{new} is such that it is not a core member of any of the C clusters, its potential is lower than all the potentials of these clusters, but is high enough to become a new cluster center.

Case iii) The potential of x_{new} is such that it is better than the potential of an existing cluster center.

The process of determining the case that is applicable to the new object x_{new} and the actions taken are described in the following steps.

1. First, the stored potentials of all previous cluster prototypes P_i are raised by the degree to which this x_{new} is close to these previously known cluster prototypes. Hence the potential of the cluster prototype will increase more if x_{new} is closer to it (as x_{new} makes that cluster more dense) compared to when x_{new} is farther away. The maximum increase

in potential is 1, i.e., when x_{new} coincides with an existing cluster center. Let these updated potentials be P'_i.

2. We check if the potential of x_{new} is better than the first cluster center, that is whether $P_{new} > P'_1$. If true, then Case (iii) holds and the actions taken are described later below. If not, we then proceed with the subtractive step, i.e., potential of x_{new} will be reduced depending on its similarity with the first cluster center. After this subtractive step, if P_{new} becomes negative, it means that x_{new} is a core element or very close to the first cluster center, and hence Case (i) holds. No further comparison or subtraction is required.

3. If the potential does not become negative, then we continue to compare the updated potential of x_{new} with the potential of next cluster prototype. Again if the potential of x_{new} is less than this cluster center, we perform the subtractive step as described above. If the potential of x_{new} is greater than the current cluster center, i.e., x_{new} has potential better than the potential of an existing cluster center, then Case (iii) holds and we continue with step 4a below; otherwise we proceed with the subtractive step. Also it might so happen that after comparison with the potentials of all existing C cluster prototypes and going through the corresponding subtractive steps, x_{new} has a potential good enough to become a new cluster center, which indicates Case (ii) holds and we continue with step 4b.

4a. If Case (iii) holds, we calculate the impact factor for not making x_{new} as one of the existing cluster centers. Computation of impact factor is discussed below. If the impact factor exceeds a threshold value, specified by a web administrator, then a complete reclustering has to be performed.

4b. If Case (ii) holds, x_{new} becomes a new cluster centre and the model now has C+1 prototypes. Additionally, the membership of all the existing objects is also calculated

with respect to this new cluster and stored in matrix u which will now have one more row to store the membership values for the newly generated cluster.

5. Irrespective of which case we have, the membership of x_{new} is calculated with all the existing clusters.

Impact Factor

Impact factor, denoted $\mathcal{I}$, is a measure of the degree to which the cluster model is affected by not making the new incoming object as one of the existing cluster centers, when its potential is such that using the regular RFSC (for clustering of the complete dataset) it would have been a cluster centre. Impact factor is calculated only when we have case (iii). We first explain the notion of impact factor with the help of an example and then provide a formal definition. Fig. 13(a) shows a clustering of N_U objects obtained from RFSC. It shows three clusters with their centers marked.

Now suppose a new object x_{new} arrives and has to be included in the existing clustering. If case (iii) becomes true, then x_{new} has potential better than one of the existing cluster centers. Two situations can arise. Fig. 13(b) illustrates the first case wherein x_{new} lies very close to an existing cluster center. Making x_{new} the cluster center of this cluster will not change much the clustering we have because there already exists a cluster center very close to x_{new} and the membership of x_{new} will be very close to 1. Hence the impact of not making this a cluster center is very low. The other situation is shown in Fig. 13(c), wherein we see that x_{new} is sufficiently far from any of the existing cluster centers. This would mean that the impact of not making this object as a cluster center will be much greater.

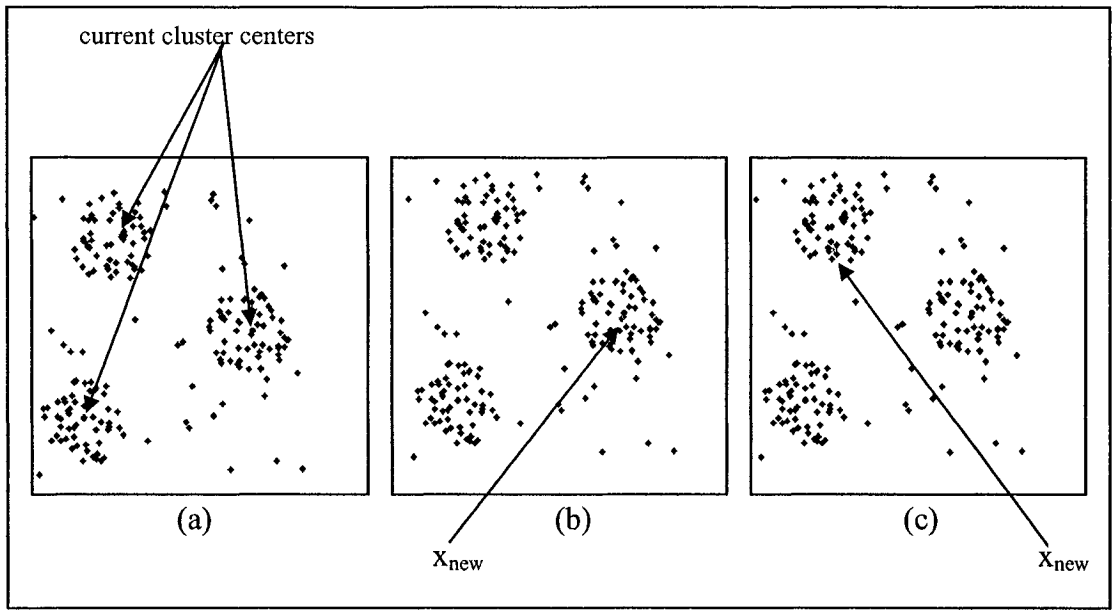


Figure 13 Example for impact factor calculation

We shall now give a quantitative measure for impact factor. The impact factor I is initialized to 0 the first time incremental RFSC algorithm is applied and also every subsequent time that RFSC is used to perform reclustering of the entire dataset. When a new object x_{new} is to be included in an existing cluster, if Case (iii) holds, we find the minimum of the dissimilarity of x_{new} to existing cluster centers and increase the impact factor. Hence if there is a cluster center near x_{new}, then the impact is less, whereas the impact is more if the cluster centers are far away. When the impact factor reaches a pre-defined threshold, reclustering of the entire dataset is required.

Our maintenance algorithm with the impact factor considered is presented below.

Algorithm incrementalRFSC

1. Calculate the potential P_{new} of the new object x_{new} to be inserted.

2. Raise the potentials of all existing cluster prototypes;
   $D_{i,\,new}$ = dissimilarity between x_{new} and i^{th} cluster center
   $P_i = P_i + e^{-\alpha D_{i,new}^2}$;
3. d_{min} = minimum of the dissimilarity values between x_{new} and all the

previously found cluster centers.
for i=1 to C **do**
if ($P_{new} > P_i$) **then**
if ($P_{new} > \overline{\in} P_1$), **then** //case (iii)
I += d_{min}
else if ($P_{new} \geq \in P_1$) **then**
If (d_{min}/γ)+(P_{new}/P_1) ≥ 1, **then** //case (iii)
I += d_{min}
end if
end if
else
$P_{new} = P_{new} - P_i\ e^{-\alpha D_{i,new}^2}$;
If ($P_{new} < 0$), **then** goto step 5;
//case (i) -- no further comparisons required
end if
end for
4. **if** (($P_{new} > 0$) and (Case (iii) is not true))
//check if x_{new} can become a new cluster center
if ($P_{new} \geq \in P_1$), **then**
If (d_{min}/γ)+(P_{new}/P_1) ≥ 1, **then**
//case (ii) is true, x_{new} is the new cluster center
C = C+1; //increment cluster count
$P_C = P_{new}$;
Calculate the membership of N_U old objects
with respect to this new cluster;
end if
end if
end if
5. Increment N_U by 1 as a new object is added;
Calculate membership of x_{new} with C clusters.

End algorithm;

5.3 Experiments and Results

In this section, again as in previous experimental studies, we first present the evaluation metrics used by us for measuring the similarity between clustering obtained by applying Incremental RFSC maintenance algorithm and the one obtained by complete reclustering. This is followed by an analysis of the results of our experiments and the values obtained for these metrics.

5.3.1 Evaluation Metrics

For similarity metrics, we have used the Rand, Corrected Rand, and Jaccard coefficients. The Rand coefficient was used for our comparative study of RFSC and ARCA as discussed earlier in Section 3.5.4. With same notations as introduced in section 3.5.4, we restate the Rand coefficient (R) formulation for easy reference and present the formulation of Jaccard (J) and Corrected Rand (CR) coefficients below.

$$R = (a + d) / \binom{n}{2} \text{ and}$$

$$J = a / (a + b + c).$$

Rand and Jaccard values are in the range [0, 1]. The maximum value of 1 may not be achievable when the two clusterings have different number of clusters. [Hubert et al., 1985] suggested use of "rand index corrected for chance", called as "corrected" Rand (CR) coefficient. Correcting a statistic for chance means normalization such that its value is 0 when the partitions are selected by chance, and is 1 when a perfect match is achieved. It is defined using the following formula:

$$CR = \left(\frac{\sum_i \sum_j \binom{n_{ij}}{2} - \left[1 / \binom{n}{2} \right] \sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2}}{\left(1/2\right)\left[\sum_j \binom{n_{.j}}{2} + \sum_i \binom{n_{i.}}{2} \right] - \left[1 / \binom{n}{2} \right] \sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2}} \right)$$

5.3.2 Similarity Analysis

We now evaluate our maintenance algorithm by comparing the partitions obtained with those obtained by complete reclustering. We use the same dataset and similarity measure

described earlier in section 3.5.1 and 3.5.2 respectively. Let N_{old} denote the number of sessions in the initial set of sessions clustered by RFSC, and N_{new} denote the number of new sessions which are to be added to the cluster model of the N_{old} sessions. Sessions are strictly ordered by time, i.e., in the order in which they were formed when users accessed the website. We created 9 experimental case studies by choosing different values of N_{old} as 30000 (46.49%), 34000 (52.68%), 38000 (58.88%), 42000 (65.08%), 46000 (71.28%), 50000(77.48%), 54000(83.68%), 58000(89.88%) and 62000 (96.08%). For each case, the number of increment N_{new} = 64529 - N_{old}. These (N_{old}, N_{new}) pairs can also be seen in Table 6.

For each case, we applied RFSC to cluster the N_{old} sessions and used the index of goodness defined in Section 3.4 to achieve the best clustering for these N_{old} sessions. We then applied our maintenance algorithm in each case to include the new N_{new} sessions into the existing clustering. For validation and comparison purposes, we also performed reclustering of the entire dataset, i.e., N_{old} + N_{new} = 64529 sessions, using index of goodness to achieve the best clustering for the entire dataset. We obtained C=46 clusters. Since Rand, CR, and Jaccard coefficients are defined for crisp clustering, we defuzzified the clusters obtained by applying the maintenance and reclustering techniques. Noise sessions, which have low membership with all the prototypes are put into an extra cluster, called noise cluster, similar to what is done in Section 3.5.4. That is, if a clustering yields C clusters, we add the $(C+1)^{th}$ cluster as the noise cluster during the defuzzication step. After defuzzification, we measure the similarity of clustering obtained from the incremental RFSC and reclustering operations. The results of these experiments for measuring similarities are shown in Fig. 14.

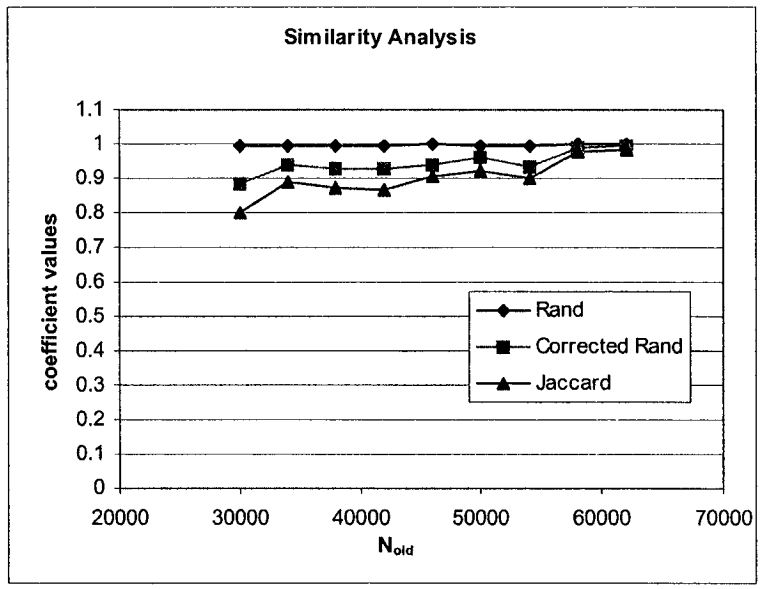


Figure 14 Similarity between incremental RFSC and reclustering

Particularly interesting is the Corrected Rand (CR) coefficient since it has the maximum value of 1 when the clustering structures (partitions) compared are identical. It has the value 0 when they are generated by chance (i.e., CR is corrected for chance). The CR values obtained are 0.883, 0.938, 0.928, 0.925, 0.937, 0.959, 0.931, 0.988 and 0.992, respectively, for the 9 cases studied in these experiments. These are very high values of CR (high similarity) noting that the first case had N_{old} = 30000, which is even less than 50% of the entire dataset under consideration. We see that increase in N_{old} or decrease in the number of newly added sessions N_{new} leads to higher similarity of the clustering generated. These results indicate that the similarity between the clustering obtained from incremental RFSC and complete reclustering did not, in a statistical sense, happen by chance. High values of CR for very small increments (0.992 for case 9 with N_{old}=62000) indicate that RFSC is stable when the dataset grows with addition of new objects -- a desirable clustering property as discussed in Section 5.1.

Table 6 shows the computation time for adding these N_{new} sessions. Although, high similarity with the cluster model obtained by complete reclustering is a necessary condition for any maintenance algorithm, this should not come at a prohibitive cost. We observed that on an average, incremental RFSC takes around 4 seconds to add 100 new sessions. While the time taken for clustering of the entire dataset of 64529 sessions is around one and a half hour; adaptation of this cluster model to accommodate 100 new sessions can be accomplished in just a few seconds. In the absence of such a maintenance algorithm a complete reclustering would have been required, clearly showing the importance of our maintenance algorithm.

N_{old}	N_{new}	time (sec)
30000	34529	3219
34000	30529	2965
38000	26529	2675
42000	22529	2358
46000	18529	1999
50000	14529	1625
54000	10529	1210
58000	6529	776
62000	2529	301

Table 6 Timings for incremental RFSC

5.3.3 Detection of New Interest Areas

While the above experiments clearly show that incremental RFSC maintains existing clusters with varying number of new sessions (N_{new}) being added, it does not demonstrate whether new interest areas will also be detected, if the newly added data has that characteristic. Indeed, incremental RFSC is very capable of detecting new interest areas. For demonstrating this, we consider the following experimental case. Clustering of the entire dataset of 64529 sessions yields C=46 clusters as discussed above. RFSC detects clusters according to its order of significance in the dataset i.e., the most prominent

cluster (interest area) is found first, followed by the next prominent cluster, and so on. We defuzzified these clusters and created 7 pairs of (N_{old}, N_{new}) partitions of the entire dataset by hiding clusters 1, 3, 5, 10, 26, 32, and 44, one at a time. These clusters were chosen at random. Hence in this process we hide the most prominent as well as lower potential clusters found by RFSC in the entire dataset. For each such partition, we apply RFSC to N_{old} sessions to get C_{old} clusters. The cardinalities of these (N_{old}, N_{new}) partitions and the values of C_{old} are shown in Table 7, in which the first column indicates the hidden cluster /profile.

Hidden Cluster	**N_{old}**	**N_{new}**	**C_{old}**	**No. of new clusters**	**Impact factor, $\mathcal{I}$**	**Corrected Rand (CR)**
1	60964	3565	47	1	1845.2	0.881
3	61799	2730	45	1	1261.7	0.949
5	62697	1832	47	0	221.7	0.885
10	62184	2345	45	1	1316.0	0.988
26	63745	784	45	1	95.7	0.998
32	64032	497	45	1	72.8	0.997
44	64033	496	45	1	68.8	0.996

Table 7 New interest areas detected with incremental RFSC

Next, we use incremental RFSC to add the hidden N_{new} sessions to the cluster model, respectively. Table 7 shows the impact factor $\mathcal{I}$. Considering that N_{new} is relatively a small addition compared to N_{old}, we note that the values for impact factor is quite high especially for the more prominent clusters that were hidden like 1,3 and 10. This is because the new cluster formed is quite prominent (significant new trend or interest among users) with respect to the existing cluster model and if $\mathcal{I}$ keeps increasing at the same rate, then a complete reclustering of the entire dataset may be required. Noteworthy is the fact that in each of the above cases, we could recover the cluster that was hidden.

Also, high CR values indicate a close similarity between the model obtained from the maintenance algorithm and the original clustering of 64529 sessions in each case. One such example is shown below. The left column in Table 8 shows the original profile 3 which was hidden in the second case above and the right shows its detection with incremental RFSC. The top 10 URLs are shown with their popularity scores in the profile. This profile is related to comp471 course offered in Winter 2005. We also see that one of the highly accessed pages by students taking this course is of the tutor of this course.

Original profile (profile 3)	Profile detected with incremental RFSC
/~comp471/main.htm - 0.941	/~comp471/main.htm - 0.865
/~comp471/ - 0.932	/~comp471/ - 0.860
/~comp471/General/ASSIGNS.htm - 0.343	/~comp471/General/ASSIGNS.htm - 0.324
/~comp471/General/NOTES.htm - 0.313	/~comp471/General/NOTES.htm - 0.294
/~comp471/General/LAB.htm - 0.245	/~comp471/General/LAB.htm - 0.243
/~comp471/outline/outline.htm - 0.145	/~ha_l/comp471.html - 0.206
/programs/ugrad/cs/comp471.shtml - 0.131	/~ha_l/ - 0.180
/~ha_l/comp471.html - 0.130	/~comp471/outline/outline.htm - 0.131
/~comp471/General/PROJECT.htm - 0.125	/programs/ugrad/cs/comp471.shtml - 0.119
/~comp471/slides/ - 0.124	/~comp471/slides/ - 0.116

Table 8 Example of profile hidden and profile recovered with incremental RFSC

We also observed a discrepancy in the near faithful recovery of clusters when we hide cluster 5. When we add cluster 5, no new cluster is formed. We tried to understand why this had happened. Profile 5 is related to a student who took comp239: Mathematics for Computer Science II and comp352: Data Structures and Algorithms together. This profile is shown in Table 9 (a). Now, in the clustering obtained by applying RFSC to N_{old} = 62697 sessions, we saw that profile 13 is related to the instructor and students of comp352 while profile 21 is related to the instructor of comp239 course and the students taking that course. Hence, the sessions that were hidden got split into these two clusters

and no new cluster was formed. This is also suggested by the low impact factor value i.e. the impact of not making this a new cluster is very low. Profiles 13 and 21 after the addition of the N_{new} =1832 sessions is shown in Table 9 (b) and (c) respectively. We only show the top 5 most popular pages.

Profile 5 (hidden)
/current_students.shtml - 0.616 /~comp239/ - 0.559 /~comp239/2005W/ - 0.552 /~comp352/ - 0.459 /~comp352/2005w/ - 0.449

(a)

Profile 13 (after maintenance)	Profile 21(after maintenance)
/~comp352/2005w/ - 0.775 /~comp352/ - 0.645 /~cc/COMP352/index.html - 0.410 /~cc/COMP352/announcements.html - 0.279 /current_students.shtml - 0.256	/~chvatal/ - 0.738 /~comp239/2005W/ - 0.706 /~chvatal/239/ - 0.556 /~comp239/ - 0.303 /programs/ugrad/cs/comp239.shtml - 0.115

(b) (c)

Table 9 Explaining discrepancy for hidden profile 5.

Another, interesting experimental case is when we hide the first cluster. Procedurally, RFSC finds subsequent clusters according to their potential with respect to the first prominent cluster. But, in this case, we are hiding the first cluster itself and then clustering the remaining sessions to obtain C_{old}=47 clusters. With addition of N_{new} =3565 using our incremental RFSC, we could recover the cluster though we see a high impact factor. Again, this is because if the entire dataset was clustered, then this interest area would have been the most prominent one.

For illustrating this visually, we also carried out some experiments using several synthetically generated datasets consisting of points in two-dimensional Euclidean space. One such dataset, shown in Fig. 15, contains 400 points. This dataset is generated randomly and has three prominent clusters among the noises. As we can see, the three

clusters are centered approximately at (20,20), (70,50), and (20,80). Intentionally, we kept the density of the two clusters centered at (70, 50) and (20,80) the same, and doubled that of the cluster centered at (20,20). Normalized Euclidean distances were computed to define the dissimilarity matrix R. We then applied the RFSC algorithm and used validity index to determine the best clustering. We obtained the least value of the index at C=3. The clusters centers found (in the order of their potentials) were (67.98, 49.25), (21.33, 78.23), and (19.37, 18.43).

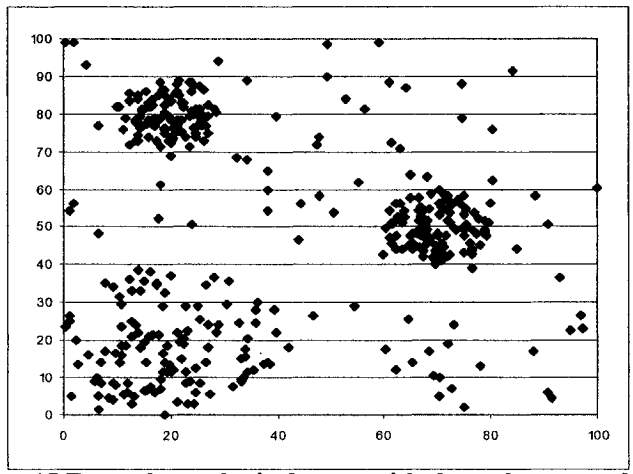


Figure 15 Example synthetic dataset with three clusters and noise

We then removed 100 points from the cluster centered at (20,80) which was found to be second most prominent cluster. This left us a dataset of 300 points with two clusters. We clustered these 300 points, using RFSC and the validity index, and found the best clustering at C=2. As a next step, we used our incremental RFSC maintenance algorithm to add the 100 points which were removed. We observed that the maintenance algorithm adds a new cluster with center at (20.39, 73.33), very close to (21.33, 78.23), which was found by reclustering of the entire dataset. Also the impact factor *I* was found to be 1.4374, which is quite low considering the fact that 100 new points were added to a dataset of 300 points. This also indicates that in most situations where Case (iii) holds, the candidate center was very close to an existing cluster center.

5.3.4 Adaptive Usage Profiling for Web Personalization

The main goal of adapting the model to the dynamic changes in the web environment is to increase the relevance and accuracy of recommendations to the user. We now discuss the results of the experiments performed to determine the recommendation quality when using the maintenance algorithm. incremental RFSC fits in well with our fuzzy hybrid CF technique introduced in Chapter 4 and hence we compare recommendation quality for the following three techniques:

1) Fuzzy hybrid CF using incremental RFSC

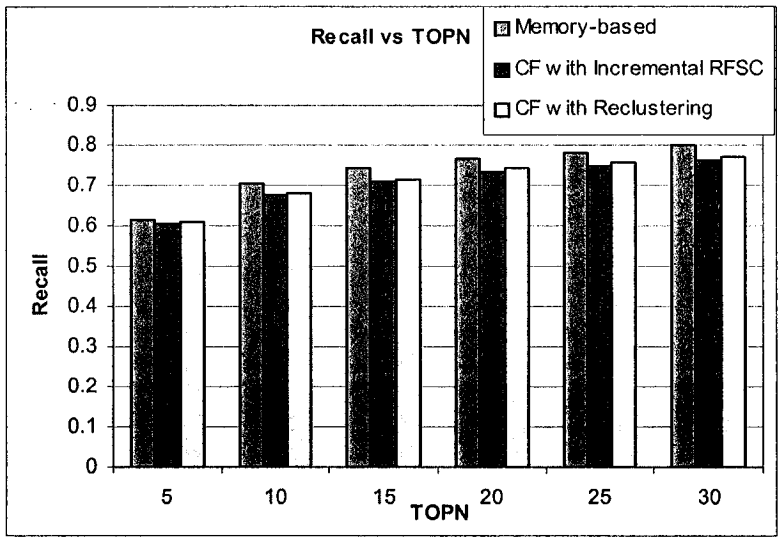
2) Fuzzy hybrid CF using Reclustering

3) Memory-based CF

For this study, we divided the dataset of 64529 sessions into two sets: the training set with 42000 sessions (65.08%), and the test set with 22529 sessions (34.92%). Again the sessions are ordered by time they are created and the division was also made considering this order. We applied the RFSC algorithm and the index of goodness. In all, 47 clusters were found, i.e., C=47. For experimentation, again we used the protocol introduced in section 4.3.2 by which we randomly selected approximately 80% of the pages in each session in the test set as seen, and tried to predict the other 20% of pages which were hidden using the above algorithms. For each of the above, we checked to see if the hidden pages are present in the top-N pages recommended by the respective algorithms. For fuzzy hybrid CF with incremental RFSC, after every 4000 new sessions that were seen from the test set, these sessions were added to the model using incremental RFSC.

Similarly for fuzzy hybrid CF with reclustering, complete reclustering was performed in steps of 4000 sessions, i.e., at 46000, 50000, 54000, 58000, and 62000.

Fig. 16 shows the comparison of recommendation effectiveness, using recall, precision, and F1. For these experiments, we kept the parameter k-nearest neighbor, k=100; fuzzy K-nearest prototype, KNP=5; and varied the parameter TOPN from 5 to 30. We saw that recommendation quality obtained using incremental RFSC and complete reclustering is almost the same. Also as TOPN increases, even though we see an increase in recall, we see that precision decreases and the overall combined effect (captured by F1 measure) decreases as well. We also observe that effectiveness of fuzzy hybrid CF approaches that of memory-based CF while achieving increased efficiency.



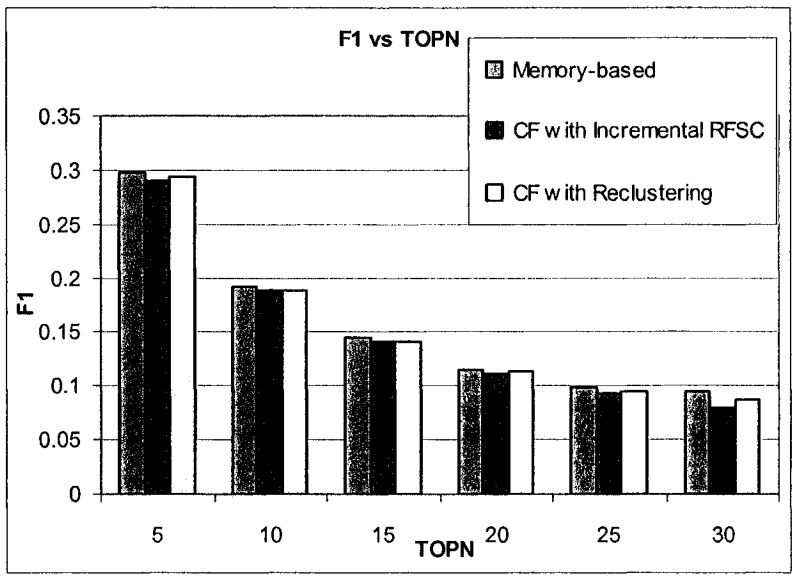


Figure 16 Comparison of recommendation effectiveness

5.3.5 Discussion

We make the following observations from results of our comprehensive experimentation.

1. To demonstrate correctness of model obtained by incremental RFSC, we compute similarity of the results from incremental RFSC clustering with the results from complete reclustering using RFSC. Our experiments show that the similarity is very high.

2. incremental RFSC is also very robust. We experimented with different batch sizes for adding new sessions ranging from 2529 to 34529 sessions. Even for truly large number of new additions, the cluster model using incremental RFSC was very similar to the reclustered model.

3. Most important of all, our experiments also show that incremental RFSC does not fail to discover new interest areas (clusters) when they ought to be found in the newly added data. We computed impact factors in each case. The values obtained

for impact factor match very well to our intuition about the newly add data. When a significantly new interest area develops and continues to evolve over time with addition of new usage data, we observe a high impact factor indicating reclustering of the entire dataset would achieve a better model.

4. Our experiments also clearly indicate that the quality of recommendation obtained using this algorithm is almost as good as complete reclustering of the entire dataset. Moreover, incremental RFSC is very fast, taking only a few seconds for adapting the usage profile model with 100 new user sessions. This is what makes it a powerful technique for online web usage profiling.

Chapter 6

Conclusions and Future Work

The World Wide Web has become an integral part of our lives being used by millions of people nowadays for a wide range of e-commerce applications. The reasons for this popularity include 24×7 availability, seamless communication over any distance, fast and cheap information access. However, further sustained growth in e-commerce mandates significant improvements in the ease and the speed with which a layman can do a business transaction on the web. Users often face the problem of information overload wherein the amount of information is far more than what can be easily assimilated and interpreted by human beings. Users generally feel lost in this huge information space. Web personalization is an approach to address this problem by providing content and services at a web site tailored to the needs of individual users from the knowledge gained through previous interactions of users with the site.

In this work, we first studied the web personalization process and the various phases of personalization. We identified the key issues of effectiveness and efficiency in current personalization systems, particularly related to recommender systems. Our first major contribution is a new algorithm for usage profile modeling. This is the Relational Fuzzy Subtractive Clustering algorithm for efficiently mining usage data of a web site to extract usage profiles which give us knowledge about the current interests and trends among the

users accessing the site. RFSC has a number of advantages over previously known fuzzy clustering algorithms. Significant among these are the following:

- scalability to large datasets,
- reduced concern over the prohibitively large model compilation time,
- no requirement for any user specified control parameters which may bias the clustering to not manifest the true structure in the data,
- high immunity to noise, which is inherent in web data, and
- ability to directly handle non-Euclidean data thus avoiding the need for the expensive β spread transformations for converting the non-Euclidean relational data into Euclidean space, that are essential for most other methods.

We also proposed a validity index called index of goodness for RFSC to measure the quality of fuzzy clustering derived along the same lines as the popular Xie-Beni index [Xie and Beni, 1991]. We conducted a comprehensive comparative study of Fuzzy C-means based Any Relational Clustering Algorithm (ARCA) algorithm and Relational Fuzzy Subtractive Clustering (RFSC). To the best of our knowledge, this is the first such comprehensive experimental study, providing us with valuable measures to objectively judge the performance of these techniques. In addition to throwing up a number of interesting observations, our study also provides a basis for future research and applications of the proposed concepts and techniques.

Exploiting these properties of RFSC gives rise to a whole new class of algorithms for effectively using and maintaining web usage profiles for different tasks. This has been amply demonstrated in our research through the development of two different improvements over collaborative filtering (CF)-based recommender systems and a very

efficient adaptive web usage profiling technique. These techniques address the two fundamental challenges in CF-based recommender systems, accuracy and scalability. The first technique called Fuzzy Hybrid CF is an improvement over cluster based recommender model. This technique achieves the accuracy of memory-based and the scalability of model-based CF, and hence inherits the advantages of both. This is achieved by making innovative use of the fuzzy prototypes and the fuzzy membership grades obtained from the RFSC model. The fuzzy nearest prototype of the active user is used to find a group of like-minded users within which a memory-based search is conducted. The second technique that we have proposed is the Two Level Model-based CF technique which is an improvement over association rule based recommender model. In this, we use RFSC as the first level modeling to separate the interest areas and then mine for association rules within individual clusters. We showed how segregation of interest areas helps in giving more relevant and pertinent recommendations to the current user.

We identified the problem of adaptive web usage profiling for dealing with dynamically changing browsing behavior of users on the web. As this has to be done online, correctness, robustness and efficiency of adaptation are crucial for the success of personalization. We proposed a profile maintenance scheme based on a new algorithm, incremental RFSC that we have devised. Our maintenance scheme consists of the following three components:

1. RFSC algorithm for initial clustering and reclustering;
2. incremental RFSC algorithm for adapting the profile model efficiently to newly arriving web usage data; and

3. An impact factor whose value can be used to decide on when reclustering is recommended.

The positive performance of this maintenance scheme as evidenced from our comprehensive experimentation has already been presented in Chapter 5. We are not aware of any other maintenance scheme that works with fuzzy clustering and provides this quality of performance.

Several directions can be exploited as a continuation of this research. We discuss some of these below.

1) **Applications and Extensions of RFSC.** In this work, we developed and successfully used RFSC in the context of mining web usage data. Our emphasis was on devising and experimentally validating a class of techniques for web personalization based on improvements to collaborative filtering-based recommender systems. We would like to step back and develop theoretical aspects of RFSC independent of the domain. What kind of other datasets can RFSC be applied to? Currently RFSC uses a Gaussian potential function to model the data. Can we use any other potential function and what would be the advantages?

   Also to obtain the best clustering we use index of goodness which finds those values of accept ratio $\overline{\in}$ and reject ratio $\underline{\in}$ for which the index is minimum. Typically values of $\overline{\in}$ and $\underline{\in}$ are varied from 0.1 to 1 in steps of 0.1, such that $\underline{\in} < \overline{\in}$ and the index of goodness is found for each combination. Could there be a more intelligent/efficient way for doing this search?

As no clustering algorithm is a universal fit for all problems, we would like to research areas such as bioinformatics and image processing, which face similar challenges like the web, and to see whether RFSC has any specific advantages over existing techniques.

Another issue is the following: Can RFSC be extended to work "Out of Core"? With the pace at which data is growing today, an out of core modeling technique will be very beneficial in future. A good start in this direction is incremental RFSC which under certain conditions could be used as an "Out of Core" algorithm. Another direction could be developing an out of core data structure to support the clustering process. Again very few works have been reported in this direction of "Out of Core" clustering.

2) **Personalized Web Search.** Current web search engines are built to serve all users, independent of the special needs of any individual user (one size fits all). As the amount of information on the web increases rapidly, it creates many new challenges for web search. When the same query is submitted by different users, a typical search engine returns the same result, regardless of who submitted the query. This may not be suitable for users with different information needs. For example, for the query "Apple", some users may be interested in documents dealing with "apple" as "fruit", while some other users may want documents related to Apple computer. Personalization of web search is to carry out retrieval for each user incorporating his/her interests [Glen and Widom, 2003, Liu et al. 2004]. For example, Google's Personalized Search allows users to specify the web page categories of interest. Some web search systems use relevance feedback

to refine user needs or ask users to register their demographic information beforehand in order to provide better service, like, Yahoo®. Typically these systems require users to engage in additional activities beyond search to specify/modify their preferences manually. Can we develop approaches that are able to implicitly capture users' information needs?

Another issue is the following: Can we improve search results by using soft computing techniques to capture fuzziness and uncertainty in the mind of the user? How can we further improve scalability and efficiency?

The above are only a few of the directions in which our current research can be carried further. Clearly, the primary technique for fuzzy clustering introduced in this thesis, Relational Fuzzy Subtractive Clustering, is abundantly rich in its properties to enable further research along many different paths and application in many different domains.

Bibliography

[Abraham, 2003] Abraham, A. Business Intelligence from Web Usage Mining, J. of Information & Knowledge Management (JIKM), World Scientific Publishing Co., 2(4), pp. 375-390.

[Adomavicius and Tuzhilin, 2005] Adomavicius, G., Tuzhilin, A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Trans. Knowl. Data Eng. 17(6): 734-749 , 2005.

[Agrawal and Srikant, 1994] Agrawal, R., Srikant, R. Algorithms for Mining Association Rules, In Proc. VLDB'94, Santiago, Sept 1994.

[Agrawal and Srikant, 1995] Agrawal, R., Srikant, R. Mining Sequential Patterns. In Proceedings of the International Conference on Data Engineering (ICDE'95), Taipei, Taiwan, March 1995.

[Breese et al., 1998] Breese, J., Heckerman, D., Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering, In Proc. of UAI-98, pp. 43-52, 1998.

[Bezdek 1981] Bezdek, J.C. Pattern recognition with fuzzy objective function algorithms, Plenum, New York, 1981.

[Burke, 2002] Burke, R. Hybrid recommender systems: Survey and experiments, User Modeling and User Adapted Interaction, 2002.

[Billsus and Pazzani, 1998] Billsus, D., Pazzani, M. Learning Collaborative Information Filters, Proc. Int'l Conf. Machine Learning, 1998.

[Billsus et al., 2002] Billsus, D., Brunk, C., Evans, C., Gladish, B., Pazzani, M. Adaptive Interfaces for Ubiquitous Web Access, Comm. ACM, vol. 45, no. 5, pp. 34-38, 2002.

[Can et al., 1987] Can, F., Ozkarahan, E.A. A dynamic cluster maintenance system for information retrieval, In Proc. of the 10th Annual International ACM-SIGIR Conference, pp. 123-131, 1987.

[Charikar et al., 2004] Charikar, M., Chekuri, C., Feder, T., Motwani, R. Incremental clustering and dynamic information retrieval, In SIAM Journal on Computing 33 (6), pp. 1417-1440, 2004.

[Chiu, 1994] Chiu, S.L. Fuzzy model identification based on cluster estimation, J. of Intelligent and Fuzzy Systems, 2(3), 1994.

[Chien and George, 1999] Chien, Y., George, E.I. A Bayesian Model for Collaborative Filtering, Proc. Seventh Int'l Workshop Artificial Intelligence and Statistics, 1999.

[Corsini et al. 2004] Corsini, P., Lazzerini, B., Marcelloni, F. A new fuzzy relational clustering algorithm based on fuzzy C-means algorithm, Soft Computing, Springer-Verlag,2004.

[Cooley, 2000] Cooley, R. Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data, PhD thesis, University of Minnesota, 2000.

[Cooley et al., 1999] Cooley, R., Mobasher, B. and Srivastava, J. Data Preparation for Mining World Wide Web Browsing Patterns, J. of Knowledge and Information Systems, 1, pp. 1-27, 1999.

[Coenen et al., 2000] Coenen, F., Swinnen, G., Vanhoof, K., Wets, G. 2000. A framework for self adaptive websites: Tactical versus strategic changes. In Proc. of

WEBKDD'2000, at Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Boston, August).

[Delgado and Ishii, 1999] Delgado, J., Ishii, N., Memory-Based Weighted-Majority Prediction for Recommender Systems, Proc. ACM SIGIR '99 Workshop Recommender Systems: Algorithms and Evaluation, 1999.

[Dave, 1991] Dave, R.N. Characterization and detection of noise in clustering, Pattern Recognition Letters, vol. 12, 657-664, 1991.

[Dave and Sen, 2002] Dave, R.N., Sen, S. Robust Fuzzy Clustering of Relational Data, IEEE Trans. Fuzzy Sys., 10(6), pp. 713-726, 2002

[Eirinaki and Vazirgiannis, 2003] Eirinaki M., Vazirgiannis M., Web mining for Web personalization, ACM Transactions on Internet Technology 3(1): 1-27, 2003.

[Ester et al., 1998] Ester, M., Kriegel, H., Sander, J., Wimmer, M., Xu, X.: Incremental clustering for mining in a data warehousing environment, In Proc. of VLDB 1998, Morgan Kaufmann Publishers Inc., pp. 323-333, 1998.

[Fu et al., 2000] Fu, X., Budzik, J., Hammond, K., Mining Navigation History for Recommendation, In Proc. Intelligent User Interfaces, New Orleans, LA, Jan 2000.

[Getoor and Sahami, 1999] Getoor, L. Sahami, M., Using Probabilistic Relational Models for Collaborative Filtering, Proc. Workshop Web Usage Analysis and User Profiling (WEBKDD '99), Aug. 1999.

[Glen and Widom, 2003] Glen, J., Widom, J. Scaling Personalized Web Search. In Proceedings International WWW Conference, Budapest, Hungary, 2003

[Hathaway and Bezdek, 1994] Hathaway, R.J., Bezdek, J.C. NERF c-means: Non-Euclidean relational fuzzy clustering, Pattern Recognition 27, pp. 429-437, 1994.

[Hathaway et al. 1996] Hathaway, R.J., Bezdek, J.C., Davenport, J.W. On relational data version of c-means algorithm, Pattern Recognition Letters 17, pp. 607-612, 1996.

[Han and Kamber, 2000] Han, J., Kamber, M., Data Mining: Concepts and Techniques, Morgan Kaufmann, 2000.

[Hofmann, 2003] Hofmann, T., "Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis," Proc. 26th Int'l ACM SIGIR Conf., 2003.

[Jain and Dubes, 1988] Jain, A. K., Dubes, R. C. Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, N. J., 1988.

[Konstan et al., 1997] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., and Riedl, J. Applying collaborative filtering to usenet news, Communications of the ACM, 40(3):77-87, 1997.

[Keller et al., 1985] Keller, J., Gray, M. and Givens, J., A fuzzy k-nearest neighbor algorithm, IEEE Transaction on Systems, Man and Cybernetics, 15(4): 580, 1985.

[Linden et al., 2003] Linden G., Smith B., and York J. Amazon.com Recommendations Item-to-item collaborative filtering, IEEE Internet Computing, 7(1), 76-80, 2003.

[Liu et al. 2004] Liu, F., Yu, C., Meng, W. Personalized Web Search for Improving Retrieval Effectiveness, IEEE Transactions on Knowledge and Data Engineering, 2004.

[Mobasher, 2004] Mobasher, B. Web Usage Mining and Personalization, Practical Handbook of Internet Computing, Munindar P. Singh (ed.), CRC Press, 2004.

[Mobasher et al., 1999] Mobasher, B., Cooley, R., Srivastava, J. Creating adaptive web sites through usage based clustering of URLs. In Proc. of KDEX'99, Nov. 1999.

[Mobasher et al., 2000] Mobasher, B., Cooley, R., Srivastava, J. Automatic personalization based on web usage mining. Comm. ACM, 43, pp. 142–151, August 2000.

[Mobasher et al., 2001] Mobasher, B., Dai, H., Luo, T., Nakagawa, M. Improving the Effectiveness of Collaborative Filtering on Anonymous Web Usage Data, In Proc. of ITWP'01, Seattle, August 2001.

[Nasraoui et al., 2005] Nasraoui, O. World Wide Web Personalization, Encyclopedia of Data Mining and Data Warehousing, J. Wang, Ed, 2005, Idea Group.

[Nasraoui et al., 2000] Nasraoui O., Frigui H., Krishnapuram R., Joshi A. Extracting Web User Profiles Using Relational Competitive Fuzzy Clustering, In Proc. International Journal on Artificial Intelligence Tools, 9(4): 509-526, 2000.

[Nasraoui et al., 2002] Nasraoui O., Krishnapuram R., Joshi A., Kamdar T. Automatic Web User Profiling and Personalization using Robust Fuzzy Relational Clustering, in E-Commerce and Intelligent Methods Ed., 2002, Springer-Verlag.

[Nasraoui et al., 2003] Nasraoui O., Cardona C., Rojas C., and Gonzalez F.: Mining Evolving User Profiles in Noisy Web Clickstream Data with a Scalable Immune System Clustering Algorithm, In Proc. of WebKDD, Washington DC, August 2003.

[Nakagawa and Mobasher, 2003] Nakagawa, M., Mobasher, B. A Hybrid Web Personalization Model Based on Site Connectivity, In Proc. of WEBKDD 2003, pp. 59-70, Washington USA, August 28, 2003.

[O'Connor and Herlocker, 1999] O'Connor, M. & Herlocker, J. Clustering Items for Collaborative Filtering, In Proc. of the ACM SIGIR Workshop on Recommender Systems, CA, 1999.

[Pennock et al., 2000] Pennock, D. M., Horvitz, E., Lawrence, S., and Giles, C. L. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach, In Proc. of UAI-2000, pp. 473-480, Stanford, CA, 2000.

[Perkowitz and Etzioni, 1998] Perkowitz, M., Etzioni, O. Adaptive Web Sites: Automatically Synthesizing Web Pages, In Proc. of 15th National Conference on Artificial Intelligence, Madison, WI, July 1998.

[Perkowitz and Etzioni 2000] Perkowitz, M., Etzioni, O. Adaptive web sites. Comm. ACM, 43, 8 (August), 152–158, 2000.

[Rijsbergen, 1979] Van Rijsbergen, C. J.: Information Retrieval. 2nd ed. Butterworths, London, 1979.

[Shani et al. 2002] Shani, G., Brafman, R., Heckerman, D. An MDP-Based Recommender System, Proc. 18th Conf. Uncertainty in Artificial Intelligence, Aug. 2002.

[Sarwar et al., 2000a] Sarwar, B., Karypis, G., Konstan, J., Riedl, J. Application of dimensionality reduction in recommender system—A case study. In Proc. of WebKDD 2000 Web Mining for e-Commerce Workshop, Boston, USA, 2000.

[Sarwar et al., 2000b] Sarwar, B. M., Karypis, G., Konstan, J. A., Riedl, J. Analysis of recommender algorithms for e-commerce. In Proc. of the 2nd ACM E-commerce Conference, Minnesota, USA, 2000.

[Sarwar et al., 2001] Sarwar, B. M., Karypis, G., Konstan, J. A., Riedl, J. Item-Based

Collaborative Filtering Recommendation Algorithms, In Proc. 10th Int'l WWW Conf., 2001.

[Schafer et al., 1999] Schafer, J.B., Konstan, J.A., Riedl, J. Recommender Systems in E-Commerce. In ACM Conference on Electronic Commerce (EC-99), pages 158-166, 1999.

[Shardanand and Maes, 1995] Shardanand, U., Maes, P. Social Information Filtering: Algorithms for Automating 'Word of Mouth', In Proc. Conf. Human Factors in Computing Systems, 1995.

[Shahabi et al., 2002] Shahabi, C., Banaei-Kashani, F. A Framework for Efficient and Anonymous Web Usage Mining Based on Client-Side Tracking. In Proc. of WEBKDD 2001, Springer-Verlag, New York, 2002.

[Shahabi et al., 2003] Shahabi, C., Chen, Y. Web Information Personalization: Challenges and Approaches, In 3rd Workshop on Databases in Networked Information Systems, Japan, 2003

[Shahabi et al., 2001] Shahabi, C., Kashani, F., Chen, Y., McLeod, D. Yoda: An Accurate and Scalable Web-Based Recommendation System. In Proc. of CoopIS 2001, pp. 418-432, Italy, 2001.

[Spiliopoulou and Faulstich, 1998] Spiliopoulou, M., Faulstich, L. WUM: A Tool for Web Utilization Analysis. In Proc. of EDBT Workshop at WebDB'98, LNCS 1590, pages 184–203. Springer Verlag, 1999.

[Srivastava et al., 2000] Srivastava, J., Cooley, R., Deshpande, M., Tan P. Web Usage Mining: Discovery and applications of usage patterns from web data. SIGKDD Explorations, 1(2), 2000.

[Suryavanshi et al. 2005a] Suryavanshi, B.S., Shiri, N., Mudur, S.P. An Efficient Technique for Mining Usage Profiles using Relational Fuzzy Subtractive Clustering", In Proc. of Int'l Workshop on Challenges in Web Information Retrieval and Integration (WIRI 05), Held at ICDE 2005, Tokyo, Japan, April 8-9, 2005

[Suryavanshi et al., 2005b] Suryavanshi, B.S., Shiri, N., Mudur, S.P. A Fuzzy Hybrid Collaborative Filtering Technique for Web Personalization", In Proc. of the Third Workshop on Intelligent Techniques for Web Personalization (ITWP 05), Held at IJCAI 2005, Edinburgh, Scotland, August, 2005

[Suryavanshi et al., 2005c] Suryavanshi, B.S., Shiri, N., Mudur, S.P. Improving the Effectiveness of Model Based Recommender Systems for Highly Sparse and Noisy Web Usage Data, In Proc. of 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), Compiègne, France, September 19-22, 2005

[Suryavanshi et al., 2005d] Suryavanshi, B.S., Shiri, N., Mudur, S.P. Incremental Relational Fuzzy Subtractive Clustering for Dynamic Web Usage Profiling, In Proc. of WebKDD '05 - Taming Evolving, Expanding and Multi-faceted Web Clickstreams, in conjunction with the ACM-SIGKDD 2005,Chicago, Illinois, August, 2005.

[Suryavanshi et al., 2006] Suryavanshi, B.S., Shiri, N., Mudur, S.P. Adaptive Web Usage Profiling, submitted for consideration in LNCS WebKDD book, Nasraoui, Zaiane, Spiliopoulou, Mobasher, Masand, and Yu (eds.), 2006.

[Tasoulis et al., 2005] Tasoulis, D., Vrahatis, M. Unsupervised Clustering on Dynamic Databases. Pattern Recognition Letters (to appear), 2005.

[Ungar and Foster, 1998] Ungar, L. H., Foster, D. P. Clustering methods for collaborative filtering. In Proc. of the 1998 Workshop on Recommender Systems, 1998, AAAI Press.

[Xie and Beni, 1991] Xie, X.L., Beni, G. A validity measure for fuzzy clustering, IEEE Transactions on PAMI, 13(8), pp. 841-847, 1991.

[Xu and Wunsch, 2005] Xu, R., Wunsch, D., II. Survey of clustering algorithms, IEEE Transactions on Neural Networks, page(s): 645- 678, Volume: 16, Issue: 3, May 2005.

[Yan et al. 1996] Yan, T. W., Jacobsen, M., Garcia-Molina, H., Dayal, U., From User Access Patterns to Dynamic Hypertext Linking, In Proc. of the Fifth International World Wide Web Conference, 1996.

[Yager and Filev, 1994] Yager R.R., Filev D.P., Approximate clustering via the mountain method. IEEE Transaction on System Man Cybern; 24:1279–1284, 1994.

[Yu et al., 2002] Yu, K., Xu, X., Tao, J., Ester, M., Kriegel, H. Instance Selection Techniques for Memory-Based Collaborative Filtering, In Proc. Second SIAM Int'l Conf. Data Mining (SDM '02), 2002.

[Zadeh, 1965] Zadeh, L. Fuzzy sets, Information Control, vol. 8, pp. 338–353, 1965.