University of Alberta

Library Release Form

Name of Author: Jia Li

Title of Thesis: Using Distinct Information Channels for a Hybrid Web Recommender System

Degree: Master of Science

Year this Degree Granted: 2004

Jia Li
Department of Computing Science
211 Athabasca Hall
Edmonton, Alberta
Canada T6G 2E1

Date: _____

University of Alberta

Using Distinct Information Channels for a Hybrid Web Recommender System

by

Jia Li

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2004

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Using Distinct Information Channels for a Hybrid Web Recommender System** submitted by Jia Li in partial fulfillment of the requirements for the degree of **Master of Science**.

_____

Dr. Osmar R. Zaïane
Supervisor

_____

Dr. Vadim Bulitko

_____

Dr. Lukasz Kurgan
External Examiner

Date: _____

To my parents

Abstract

Web recommender systems anticipate the information needs of on-line users and provide them with recommendations to facilitate and personalize their navigation. A variety of approaches have been attempted and proposed. Among them, using Web access logs to generate users' navigational profiles for recommendation is a popular one, given its non-intrusiveness. However, using only one information channel, namely the Web access history, is often insufficient for accurate recommendation prediction. In this thesis, we advocate the use of additional information channels, such as the content of visited pages and the connectivity between Web resources, to better generate the user profile and to build a hybrid recommender system. We test and evaluate our framework with the University of Alberta Computing Science Department Web site data. Our experiments show that this system can significantly improve the quality of Web site recommendation by combining these distinct information channels. In addition, we expand our approach to the context where pages are not content-rich, or content data are absent. We also test our system in an idiosyncratic data set provided by a commercial system – VIVIDESK.

Acknowledgements

I would like to express my gratitude to many people for making this thesis possible.

First and foremost, my deepest thanks to my supervisor, Dr. Osmar R. Zaïane for his invaluable advice and support. His guidance is always insightful, and his patience is highly appreciated. As an advisor, he shares with me his knowledge in this field, but his encouragement goes far beyond it. For the last few years, I have known Osmar, not only as an excellent supervisor, but also as a good friend, with whom I discuss everything. I am sure he will always be my mentor, for both my academic career and my personal life.

I owe the deepest gratitude to my parents for their deep love. As a son, I am really profoundly indebted to them. I hope I will make them proud of my achievements, as I am proud of them.

I would like to thank Dr. Robert Hayward for providing me with the VIVIDESK dataset, which has been useful and instrumental in devising my approaches. I would also like to thank him for the informative meetings regarding my project.

I am indebted to Dr. Vadim Bulitko and Dr. Lukasz Kurgan for serving as examiners of my thesis, and providing me with many valuable comments and suggestions. Other people have also helped and contributed their time to reading and commenting on this thesis. My sincere thanks go to Huiqing Li and Yu Liao.

I wish to express my deep gratitude to the faculty in our department, especially Dr. Russell Greiner, Dr. Davood Rafiei, and Dr. Janelle Harms, for their great teaching and willingness to help. My appreciation also goes to my fellow students and friends in the department, with particular acknowledgement to Dr. Tong Zheng and Tingshao Zhu, for sharing their knowledge and for many constructive discussions.

I am also fortunate to have come to know many wonderful people outside the Department of Computing Science, during my Master's study. I would like to take this opportunity to thank Dr. Elizabeth Richards, Bev Sawatzky, Dr. Don Sawatzky, Pauline Drinkwater, Jim Drinkwater, Dr. John Clark, among others, for their friendship and continuous moral support.

This research has been funded by the Alberta Ingenuity Funds[1], Alberta

[1] http://www.albertaingenuity.ca

Informatics Circle of Research Excellence[2], and the Alberta Government. The author thanks them for all their generous support. I am also grateful to the Department of Computing Science at the University of Alberta for providing an excellent work environment.

[2]http://www.icore.ca

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Research Description

We are living in the "Information Age", in which human beings produce and publish far more data than at any other period in history. With the belief that information leads to power and success, and with the assistance of sophisticated technologies, such as computers, we are also able to collect and access the enormous amounts of information available. Although this solves the problem of lack of necessary information suffered by our ancestors, we are now dealing with the problem of information overload. There are too many messages, too many journal articles, too many movies, too much content. In a word, we now have far more information than we can handle.

The World-Wide Web (WWW or the Web) serves as an illustration of this problem. The Web was originally created to make the storage, publishing, and delivery of information easier. Ironically, people are finding it increasingly frustrating to attempt to locate and access on-line resources. In [107], the author provides a good summary of the underlying reasons:

- The Web is extremely large. A study [88] shows that there are more than 10 billion unique, publicly accessible pages on the Web. Moreover, approximately 6 terabytes of new content is added to the Web every month;

- Web data changes rapidly. While the Web grows quickly in size, the information it contains is also updated constantly. According to [33],

the average lifetime of a Web document is 75 days, and approximately 600GB of Web data changes every month;

- The Web is poorly organized. Although small sections of the Web may be well structured and maintained, the Web as a whole is highly unstructured. Web resources are published and distributed in an uncontrolled manner, and there is no standard mechanism to guarantee the locating of existing Web resources;

- The Web user community is very diverse. On-line users in different communities may have different backgrounds, interests, and preferences. In addition, a particular user community may be interested in only a very small portion of the Web.

As a result of the above, on-line users have increasing difficulty in locating the right on-line information at the right time [61]. Most Web users have had the experience of taking an hour or more to find a Web document that they can go through in five minutes. The amount of on-line information vastly outstrips any individual's capability to survey it; and how to find desired information efficiently and effectively has become a increasingly important and emergent issue for the cyber community.

Many attempts have been made to provide tools to assist people in accessing on-line information. The search engine, Google[1] as an example, has become an inherent component of the Web, helping users to pinpoint relevant resources. The *Adaptive Website* technique [74] attempts to adjust and improve the organization and presentation of Web sites, on-the-fly, according to visitors' preferences. A number of visualization tools [19] [16] have also been developed to facilitate users' on-line experience. The Web recommender system is another approach, which we address in this thesis.

Generally speaking, a Web recommender system attempts to predict user preferences from user data, and/or user access data, for the purpose of facilitating and personalizing users' experience on-line by providing them with

[1] http://www.google.com

recommendation lists of suggested items. The recommended items could be products, such as books, movies, and music CDs, or on-line resources, such as Web pages or on-line activities. Correct recommendation can save users considerable time and effort in locating their information needs, by freeing them from needless searching, and helping them find their interesting information quickly. Thus, the Web recommender system is expected to become as popular and common as search engines in assisting browsing on the Web.

However, building a "correct" recommender system is a challenging task, due not only to the huge and constantly changing Web, but also to the extremely diverse user community, as mentioned above. Although Web recommender systems have been extensively explored in the Web Mining and Machine Learning fields [27] [78] [43] [31] [92] [25], and some commercialized systems have been produced, such as the ones used in Amazon.com[2] and Expedia.com[3], the quality of the recommendations and the user satisfaction with such systems are still not optimal [30]. In this thesis, we make an effort to design a novel Web recommender framework to improve Web site recommendation. Being aware that on-line users prefer to surf Web sites without intrusiveness and interruption, we try to exempt any explicit user input (e.g., previous customers' rating/ranking of products) from our framework design. Rather, our system relies mainly on the Web access log to derive user navigational models for recommendation. Most web servers have access logs, to record the user's browsing and activity history, which contain much hidden information regarding users and their navigation. These access logs could provide a good alternative to explicit user ratings or feedback in deriving user navigational models. However, the information recorded in the access log is incomplete and may be even incorrect, which means a Web recommender system depending solely on that information has several problems. On the other hand, we recognize that there may be additional information channels available on the Web – such as textual content and connectivity information of Web pages – which also do not require user input. Therefore, we advocate the

[2]http://www.amazon.com
[3]http://www.expedia.com

use of these additional available information channels to better model user behavior for accurate recommendation. In this thesis, we investigate building a framework for a hybrid Web recommender system, which attempts to combine and make full use of all the information available to complement the deficiency of access data, and to improve recommendation quality.

Discovering ways to combine these distinct information channels, in turn, is worth further exploration. First, utilizing more information does not guarantee a more desirable result. Second, using more than one information channel may require additional computational cost. Determining how to use and combine information in an efficient way, so that hybridizing does not jeopardize the efficiency of the system, is therefore another issue to be considered.

1.2 Proposed Thesis

In this thesis, we investigate building a framework for a hybrid Web recommender system, which attempts to combine and make full use of three distinct information channels – the Web access log, the content of Web pages, and Web connectivity information – to achieve high-quality recommendation. In this framework, the content of web pages is used to help identify concurrent information needs of users from the access log. Users may have different goals during a visit, and they also may pursue these goals concurrently. However, this fact has so far been ignored by the Web recommender system research community. We combine content and usage information to identify and organize these concurrent information needs into what we call "missions". The identified missions are used to generate user navigational profiles. After that, linkage information is used to improve these user profiles. The connectivity is also applied to rank recommendation candidates for suitable presentation. Our experiments have proved that our approach for hybridizing significantly improves the quality of Web site recommendation. The recommendation quality is measured by how many given recommendations are correct, as well as how many clicks the recommendation allows users to skip. Our experiments also show that each additional channel used contributes to the improvement.

All combinations of the different information channels in our system are done off-line to maintain and improve system efficiency.

1.3 Contributions of the Thesis

The major contributions of this thesis are summarized as follows:

- We design a framework for a hybrid Web recommender system, which combines and makes full use of different available channels to improve recommendation quality;

- To the best of our knowledge, this is the first time that three distinct information channels – usage, content, and structure data – are used and combined to build a Web recommender system. In particular, we make extensive use of the structure data to improve the performance of the recommender system;

- The combination of the different information channels is done off-line to improve system efficiency. As far as we know, this is the first hybrid Web recommender system which does all combination work off-line;

- We propose a novel notion, *mission*, to capture users' concurrent information needs during on-line navigation. A user may pursue more than one information need during a visit; a mission is a sub-session of the visit for a given unique information need;

- Based on the notion of *mission*, a new on-line navigational model – a mission-based model – is proposed. The mission-based model has been proved superior to the previous transaction-based models in capturing users' on-line behavior for the purpose of fulfilling information needs.

- We propose a new recommendation evaluation approach, which uses the combination of two metrics – *recommendation accuracy* and *shortcut gain* – to evaluate the recommendation quality of a recommender system. The former measures the predictability of a Web recommender system,

while the latter measures to what extent accurate recommendations help a user to skip, that is, reach their goals faster.

1.4 Organization of the Thesis

This thesis is organized as follows. Chapter 2 introduces related work, including Web Mining, Web recommender systems, and tools to build recommender systems. In Chapter 3, we describe the design of a hybrid recommendation framework. We first present the overall architecture of our system, then two major modules in the framework: an off-line module and an on-line module are discussed, step by step. Chapter 4 evaluates the quality of our system and compares it with other systems, as well as validating the contribution of the different information channels for recommendation improvement. First, different possible evaluation methodologies and metrics are reviewed. Based on this discussion, we identify and propose suitable evaluation methodology and metrics to test and evaluate our system. We then illustrate and discuss our experimental results. In Chapter 5, we discuss the expansion, application, and evaluation of our framework on the VIVIDESK data. Finally, Chapter 6 concludes the thesis and discusses future work.

Chapter 2

Related Work

In this thesis, we investigate the building of a framework of a hybrid Web recommender system for website navigation. In essence, it is an improved Web usage recommender system, in that it mainly relies on Web server access log data to build users' models. However, it also complements this Web Usage data with other information channels available – such as content and structure data – to improve its recommendation quality. In this section, we review the related work to this thesis, including Web mining techniques, Web recommender systems, and Web usage based recommender systems tools.

2.1 Web Mining

In this section, we go through the basic concepts of Web mining, normally categorized into Web content mining, Web structure mining, and Web usage mining, with an emphasis on their applications to Web Recommender systems.

2.1.1 Knowledge Discovery in Databases and Data Mining

Knowledge Discovery in Databases (KDD) is defined as the process of automatic extraction of implicit, previously unknown, and potentially useful information from data in large databases [76]. For effective and efficient discovery of knowledge for large datasets, many steps are involved – including data selection and preprocessing, data transformation and reduction, data mining and pattern discovery and, lastly, post-processing and interpretation. Figure

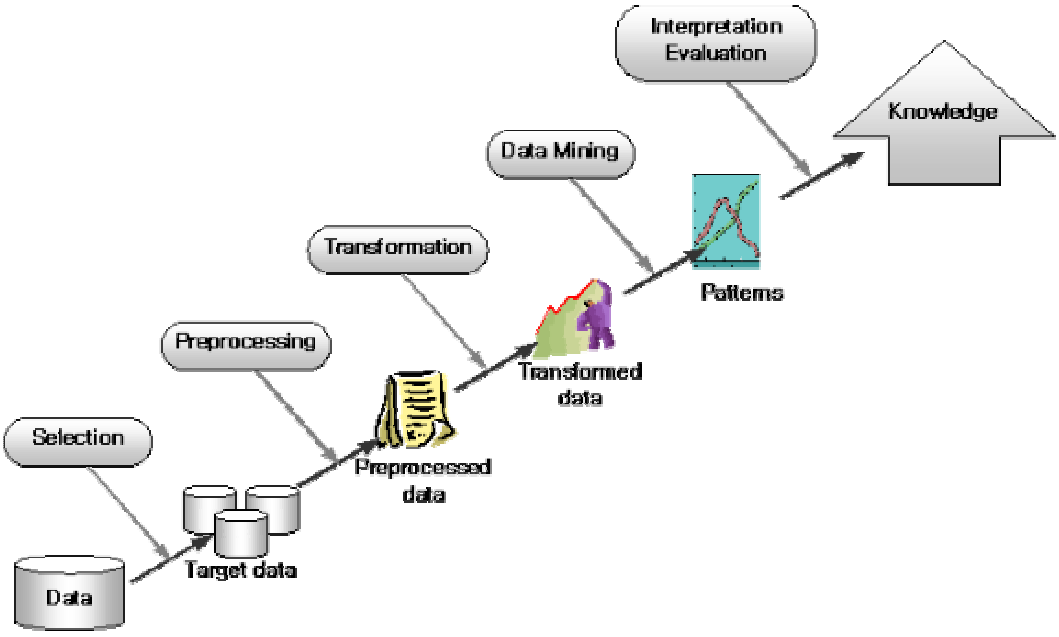

Figure 2.1: Data mining is the core of the knowledge discovery process [76]

2.1 illustrates the core steps of this process [23] [76]. Here, we give a brief introduction of each step in Figure 2.1, in turn.

- **Data Selection**: This is the first phase of any KDD process, in which data relevant to the analysis are decided on and selected from the raw data collection. This step may also include data integration in which multiple data sources, often heterogeneous, are combined in a common source;

- **Data Preprocessing**: In this phase, noise data are removed from the selected dataset. More importantly, the selected data are reconfigured and complemented to make the application of mining techniques and algorithms on them possible and effective;

- **Data Transformation**: In this step, the data are not merely transferred across but transformed, in which data are transformed into forms appropriate to the mining procedure and algorithm. This phase is treated as a sub-phase of Data Preprocessing in some KDD systems;

- **Data Mining**: This is the crucial step in which data mining techniques,

such as the *association rule* technique and *clustering* analysis, are applied in order to extract patterns previously unknown but potentially useful. Therefore, this phase is also referred to as the **Pattern Discovery** phase;

- **Interpretation and Evaluation**: This is the final phase of the KDD process, in which the discovered patterns are interpreted into knowledge and represented to the user.

From this overview, we can see that Data Mining is only one part of the knowledge discovery process. Because it is crucial in the whole process, however, **Data Mining** and **Knowledge Discovery in Databases** are frequently treated as synonyms.

2.1.2 Web Mining

Web Mining, or Data Mining on the Web, is the application of Data Mining techniques to the World-Wide Web. Traditionally, Data Mining has been applied to databases. However, the wide dissemination of the WWW technology, and the large number of document collections on the Web, have encouraged researchers to apply Data Mining to the Web. Zaïane, in his Ph.D. thesis [102], defines Web Mining as the extraction of interesting and potentially useful patterns and implicit information from artifacts or activity related to the World-Wide Web.

In [102], Zaïane also classifies Web Mining into three domains: Web Content Mining, Web Structure Mining, and Web Usage Mining. Conceptually, the Web mainly comprises three major components: the content of the Web, which encompasses the documents available; the structure of the Web, which covers the hyperlinks and relationships between Web pages/documents; and the usage of the Web, describing how and when the resources are accessed. Consequently, **Web Content Mining** is the process of extracting knowledge from the content of on-line documents or their descriptions; **Web Structure Mining** is the process of inferring knowledge from Web organization and the links between references and referents on the web; and **Web Usage Mining**,

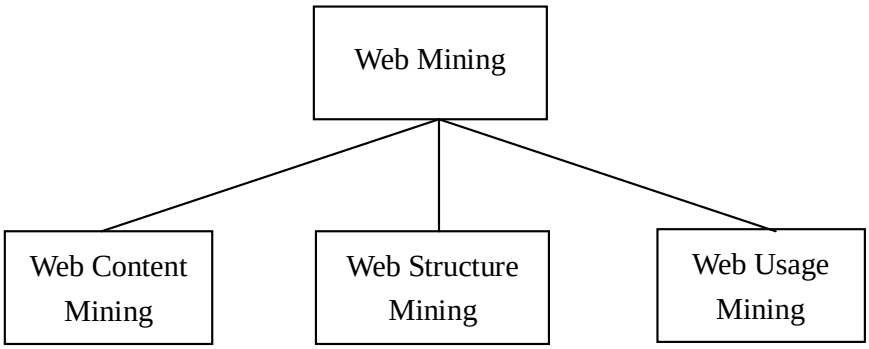

Figure 2.2: Taxonomy of Web Mining techniques [102]

also known as **Web Log Mining**, is the process of extracting interesting patterns from Web access logs, which collect and record users' on-line behavior. Figure 2.2 illustrates the taxonomy of Web Mining techniques [102].

2.1.3 Web Content Mining

The World-Wide Web and the Internet have become the biggest data repository ever built. An enormous number of authors and publishers are continuously contributing to its growth by adding all types of documents, such as text, audio, video, raw data, and so on. With the tremendous number of documents accumulated and available on-line, it has become more and more difficult for people to locate interesting and useful content from the Web. Web Content Mining, which aims at extracting relevant knowledge from documents – thereby alleviating the need to go through the retrieved documents manually in the search for pertinent knowledge – has therefore been extensively explored. Among the many kinds of on-line documents, text documents were the earliest and are still the dominant format. In fact, one study [94] indicated that 80% of a company's data is contained in text documents. Thus, Web Content Mining has so far primarily focused on and been extensively applied to text documents, and is referred to as **Web Text Mining**. Generally speaking, a framework for Text Mining consists of two components [94]: *text refining*, that transforms free-form text documents into an intermediate form; and *knowledge distillation*, which deduces patterns or knowledge from the intermediate form. The purpose of text refining is to abstract and represent text documents

in a concise form to facilitate knowledge distillation effectively, and the most commonly used text refining technique is *keywording*. Keywording scrutinizes text document content to investigate syntactical correlation and semantic association between words/terms. Relevant key words/phrases are extracted to represent the document in a feature vector $(d_1, d_2, ..., d_n)$, where d_i is the *i-th* feature and n is the total number of features. Typically, each feature in the vector corresponds to a keyword, and each entry of the vector stores a numeric weight for the corresponding feature of the document. The criteria of keywording is that the original document can be represented equally well [57]. If the selected features cannot accurately represent the original documents, any Text Mining task would be destined to fail. Thus, feature extraction becomes the major concern for Web Text Mining. [81] introduces a commonly used feature extraction method, in which the weight of each feature is the number of occurrences of particular keywords in the documents (*term frequency*). Another popular method, *TF-IDF*, combines term frequency with the inverse document frequency. For a keyword T_j, its *document frequency* df_j is defined as the number of documents in a collection of N documents in which it occurs. Suppose the term frequency of T_j in document $D_i (i = 1, ..., n)$ is tf_{ij}, then the weight of feature T_j in the document D_i is given by $W_{ij} = tf_{ij} \times \log N/df_j$. One study [98] reveals, however, that the above commonly used methods are not suitable to Web documents. Some feature extraction approaches for Web documents, therefore, have been proposed [98] [57] [56], each of which may still be designed and suitable for specific contexts [98]. Knowledge distillation, on the other hand, is generally defined to deduce patterns and relationship across documents. Document categorization, which organizes a document collection in groups, is a typical example of knowledge distillation. Document categorization can be further divided into two forms: *classification* and *clustering*: *Classification* organizes a collection of documents by predefined themes. Given a set of positive and negative training examples, a classifier is trained and can then class documents into one or more classes [42] [45]. On the other hand, *clustering* searches for predominant themes in a collection of documents and performs the categorization of all documents in the found themes [102] [44]

[105].

Web Content Mining techniques on Web pages are useful for a recommender system, in that users often visit a Web site to fulfill some information needs, while Web resources related to a given goal are generally content coherent. However, there is no guarantee that pages with similar content are related to a goal.

Nowadays, increasing amounts of multimedia data are published on-line. **Web Multimedia Mining** is utilized to mine the high-level information and knowledge from these online multimedia sources, and has recently gained the attention of many researchers [102] [40]. However, this research area is still in its infancy and considerably more work needs to be done on it. We do not consider multimedia data in this thesis.

Search Result Mining

The above discussion is concerned with the direct mining of the content of on-line documents/Web pages. There is another group of Web Content Mining strategies: search result mining [102].

Search Result Mining refers to mining the content of search results returned by other tools, most commonly, search engines. Usually, a search engine relies on keyword matching and returns a large number of documents. Search Result Mining aims at mining subsets of the returned data to refine the search results. The system presented in [54] accesses the documents retrieved by search engines, and collects information from within the document or from the data usually provided by servers, such as the URL, title, content type, content length, modification date, and links. The information is used to refine search results by retrieving pertinent documents from within the search result. [105] presents a technique for clustering documents retrieved by a set of search engines, categorizing the retrieved documents into clusters. The clusters can present overlapping, representing a higher-level view on top of the list of retrieved documents and facilitating the sifting through of the often very large search engine result list. Thus, Search Result Mining can be thought of as, to some extent, one of the precursors of Web recommendation.

2.1.4 Web Structure Mining

Thanks to the interconnections between on-line documents, the World-Wide Web can reveal more information than just that contained in documents. For example, links pointing to a document (*in-degree* links) indicate the popularity of the document, while links coming out of a document (*out-degree* links) indicate the richness or perhaps the variety of topics covered in the document. This can be compared to bibliographical citations: when a paper is cited often, it is considered to be important.

Web Structure Mining is the research field focused on analyzing the link structure of the Web. The goal of Web Structure Mining is to generate a structural summary of the Web page, the Web site, or even the whole Web. If Web Content Mining is viewed as focusing on the structure and relationship of the inner document, Web Structure Mining attempts to discover the link structure of the hyperlinks at the inter-document level.

The Web is a Graph

Apart from the content of documents, the Web provides additional information through the manner in which different documents are connected to each other via hyperlinks. The web may be viewed as a (directed) graph whose nodes are the documents and whose edges are the hyperlinks between. In [10], Broder et al. crawled and stored 203 million URLs and 1,466 million links. The primary result they achieved is an analysis of the structure of the web graph which, according to them, looks like a giant bow tie, with a strongly connected core component (SCC) of 56 million pages in the middle, and two components with 44 million pages each on the sides – one containing pages from which the SCC can be reached (the IN set), and the other containing pages that can be reached from the SCC (the OUT set). In addition, there are "tubes" that allow the OUT set to be reached from the IN set without passing through the SCC; and many "tendrils", that lead out of the IN set or into the OUT set with connecting to other components. Finally, there are also several smaller components that cannot be reached from any point in this structure.

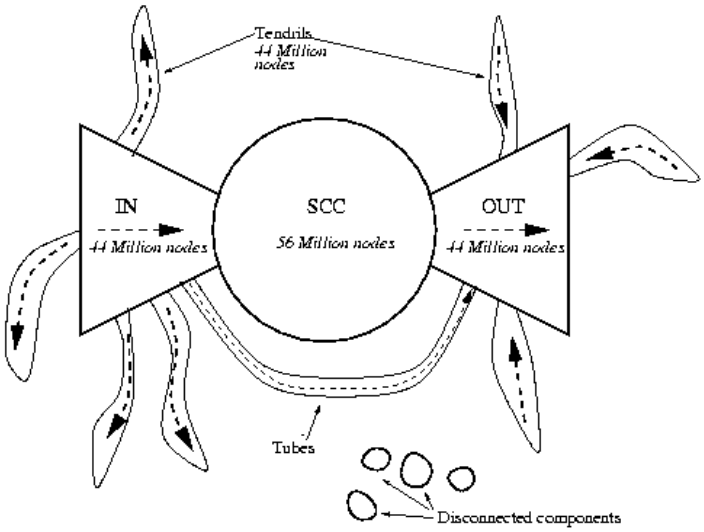

Figure 2.3: The Web is a Graph [10]

A diagram illustrating this structure is depicted in Figure 2.3 [10].

The Importance of Hyperlinks between Pages

Pages in the Web graph are inter-linked via so-called *hyperlinks*. A *hyperlink* is used to link one page (*predecessor* page) to another (*target* page or *successor* page), which may contain information helpful in knowledge discovery, and which content alone cannot provide. We try to motivate the information in the following:

- **Synonymous, Irrelevant or Misleading Text**: Words and phrases sometimes have more than one meaning; multiple topics may be covered in the same page; a document may also contain irrelevant text. All of these may lead a content-based mining system to fail. However, a hyperlink $p -> q$ may reflect the fact that pages p and q share a common topic, and that the author of page p thinks highly of q's content. Thus, such linkage could be very helpful for certain mining tasks;

- **Sparse or Non-existing Text**: Many Web pages contain only a limited amount of text. In fact, many pages contain only images and no machine-readable text at all. Looking at connectivity information would complement the deficiency of content for knowledge discovery;

- **Independent Encoding**: A link to a page (p) may originate from outside the control of the author of p (as opposed to the content of page p is under complete control of p's author). Therefore, the information provided by linkage is less sensitive to the vocabulary used by one particular author;

- **Redundancy**: Often there is more than one page pointing to a single page on the Web. The ability to combine multiple, independent sources of information can improve accuracy for certain mining tasks.

Making Use of the Linkage Information

The importance of information contained in hyperlinks has long been recognized. Anchor texts – that is, texts on hyperlinks in an HTML document of predecessor pages – were indexed by the World-Wide Web Worm [53], one of the first search engines and web crawlers. Spertus [90] suggested a taxonomy of different types of hyperlinks that can be found on the Web, and discussed how the links could be exploited for various information retrieval tasks. However, the main break-through was not made until the realization that the popularity, and hence the importance of a Web page is, to some extent, correlated to the number of in-degree and/or out-degree links, and that this information can be advantageously used for sorting and filtering the query results of a (keyword based) search engine. The in-degree of a page was first adopted to measure the importance of a page. However, in-degree alone is a poor measure because many pages are frequently pointed to without being connected to the content of the referring page (e.g., most pages in a Web site have a link that points to the home page). Therefore, more sophisticated measures are needed.

PageRank proposed by Brin and Page [9] is the one of the most successful of such measures. Specifically, PageRank is a numeric value that represents how important or authoritative a page is on the Web. When computing the PageRank of a page, not only the number of pages pointing to it (in-degrees), but also the importance of pages linking to it, are taken into account. In Brin and Page's view, when one page links to another page, it is effectively

casting a vote for the other page: the more votes that are cast for a page, the more important the page must be. Moreover, votes cast by pages that are themselves "important" weigh more heavily and help to make other pages "important."

To calculate the PageRank for a page, Brin and Page use a random surfer model. They consider PageRank as a model of user behavior, where a surfer clicks on links at random with no regard for content. As a result, PageRank of any page A can be computed as follows;

$$PR(A) = (1-d)1/n + d\sum_{1}^{n}(PR(T_i)/C(T_i)) \tag{2.1}$$

where

$PR(A)$ is the PageRank of page A;

$PR(T_i)$ is the PageRank of pages T_i which link to page A;

$C(T_i)$ is the number of out-degree links on page T_i and;

d is a damping factor which can be set between 0 and 1.

From Formula 2.1, we can see that the PageRank of a page is computed iteratively. The first term of this formula models the behavior where a surfer gets bored (with probability $(1-d)$) and jumps to a randomly selected page of the entire set of n pages. The second term uniformly distributes the PageRank of the current page to all its successor pages. Thus, a page receives a high PageRank if it is linked to by many pages, which in turn have a high PageRank and/or only few successor pages.

PageRank is the basis of the most successful search engine – Google. In Google, PageRank – a pure Web linkage/structure analysis algorithm – is combined with the textual content information of Web pages to provide search results. In general, when a user submits a query, Google searches all pages containing the keyword(s) in the query. The resulting pages are ranked according to their PageRank scores, which have been pre-computed. The higher its PageRank value, the earlier a page is presented to the user. Traditionally, a search engine can be viewed as an application of Information Retrieval with the focus on "matching": a search engine is supposed to return all those pages that match users' query, ranked by degree of match. On the other hand, the

semantics of a recommender system refers to "interesting and useful". However, Google blurs this distinction by incorporating PageRank into its ranking, which uses Web structure information to measure the authoritativeness or importance of Web pages. From this point, Google can be viewed as a form of hybrid recommender system, combining content and structure analysis with a one-input interface. (In contrast, regular recommender systems have a zero-input interface). The success of the Google system encourages us to embed Web structure analysis into our recommender system which, to the best of our knowledge, has not yet been done.

In the PageRank algorithm, all pages and all hyperlinks are treated equally. The contribution of any link to the page that it links to is the same, without consideration of the differences among links. However, the pages and links on the Web bear the following characteristics [26]:

- Some pages are informative, but some are used only for navigation. Judging whether a page is "important" or "authoritative" is only meaningful for those informative pages;

- "Authoritative" pages may not inter-link with each other as expected, but may still be pointed to by other pages. For instance, it is impossible for Honda and Ford, two fierce competitors, to link their homepages to each other, even though they are preferred by a car buyer. However, such "authoritative" pages for cars may be pointed to by other pages, e.g., a car lover's personal page.

Thus, equalizing the contribution of any page to the identification of authoritative pages is sometimes problematic. As a response, J. Kleinberg proposes two types of web pages in his study to identify authoritative pages for general search topics [36]: *Authorities* are pages that contain useful information about the query topic (which is similar to the pages with high PageRank scores), while *Hubs* are pages that link to many related *Authorities*. Based on observation of the *mutually reinforcing relationship* between Authorities and Hubs – that is, a good Hub is a page that points to many good Authorities; a good Authority is a page that is pointed to by many good Hubs – J. Kleinberg

develops an algorithm called the Hypertext-Induced Topic Selection (*HITS*) algorithm to identify both types of pages iteratively and simultaneously, as follows:

$$H_{i+1}(x) = \sum_{(x,s)} A_i(s) \tag{2.2}$$

$$A_{i+1}(x) = \sum_{(p,x)} H_i(p) \tag{2.3}$$

where

$H(x)$ represents the Hub score of the page x;

$A(x)$ represents the Authority score of the page x;

(x, y) denotes that there is a hyperlink from page x to page y.

In [36], the HITS algorithm is conducted on a so-called *focused subgraph* of the Web. A *Focused Subgraph* is defined as a collection of pages (S_σ) with the following properties:

1. S_σ is rich in relevant pages for a topic;
2. S_σ is relatively small;
3. S_σ contains most (or many) of the strongest authoritative pages of the topic.

The purpose of restricting the computation in a focused subgraph is to achieve a balance between computational effort and computational efficiency. (1) and (3) ensure the finding of good authoritative pages, as these are likely to be heavily referenced within S_σ, while (2) makes it easier to afford the computational cost of applying non-trivial algorithms. In the HITS algorithm, the construction of a focused subgraph is divided into two steps, with the assumption that a topic is specified by a query string σ. For the query σ, the algorithm first collects t highest-ranked pages for it from a keyword-based search engine. These t pages are referred to as the *root set* R_σ. The root set satisfies (1) and (2) in the list above, but is generally far from satisfying (3) [36]. The root set is then augmented to include any page pointed to by a page in it, and any page that points to a page in it. The resulting set, which is referred to as the

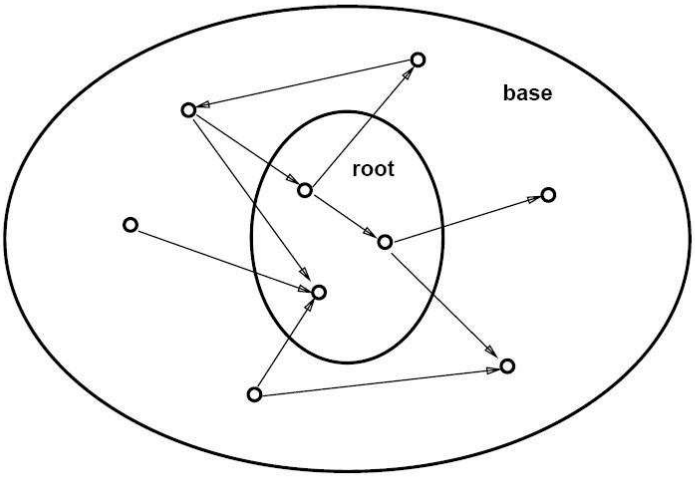

Figure 2.4: The Construction of the Focused Subgraph [36]

base set for σ, is used as the focused subgraph for σ. Figure 2.4 illustrates the construction of the focused subgraph.

When running the HITS algorithm, the Hub and Authority scores are initialized uniformly with $A_0(x) = H_0(x) = 1.0$, and normalized so that they sum up to one before each iteration. J. Kleinberg has proven that the algorithm will always converge [36], and practical experience shows that it will typically do so within a few iterations [14]. HITS has been widely used for identifying relevant documents for topics in Web catalogues [14] [5] and for implementing a "Related Pages" functionality [22], but has never been used in Web recommender systems. Our system, however, will utilize the idea of Authority and Hub to compute the importance of pages for the purpose of ranking recommendation candidates.

One of the main drawbacks of the HITS algorithm is that the Hubs and Authorities must be computed iteratively from the query result, which does not meet the real-time constraints of an on-line search engine. However, when we consider whether to use Hub/Authority or PageRank in our system, we prefer the former because it captures the Web property more accurately, as discussed above. However, because our system will perform the structure analysis in the off-line phase, there is no such real-time constraint.

Another major difficulty with the HITS algorithm is the *topic drift* prob-

lem. A study conducted by K. Bharat and M. R. Henzinger [5] reveals that a base set used in the original HITS algorithm often includes documents not relevant to the query topic. As a result, if these pages are well connected, the *topic drift* problem arises: the most highly ranked Authorities and Hubs tend not to be about the original topic [5]. One of the examples given by the authors is when running the HITS algorithm on the query "jaguar and car", the computation drifted to the general topic "car" and returned the home pages of different car manufacturers as top Authorities, and lists of car manufacturers as the best Hubs. K. Bharat and M. R. Henzinger then propose an improved HITS algorithm [5], which combines content analysis with connectivity analysis to tackle this topic drift problem. The content analysis is used to identify and eliminate non-relevant pages from the base set. More specifically, a relevance weight with the query topic is computed for each page in the base set, and all pages whose relevance weight is below a threshold are pruned. As the query topic is usually broad and general, matching the query against a document is not sufficient to measure the similarity between the document and the query topic. Instead, the authors use all the documents in the root set to re-define the query topic, and then match every document in the base set with it. Specifically, they consider the concatenation of the first 1000 words from each document to be the representation of the query topic. After that, the similarity of the query topic (Q) and any document (D), $Similarity(Q, D)$ is computed as follows:

$$Similarity(Q, D) = \frac{\sum_{i=1}^{t}(\omega_{iq} \times \omega_{id})}{\sqrt{\sum_{i=1}^{t}(\omega_{iq})^2 \times \sum_{i=1}^{t}(\omega_{id})^2}}$$

where

$\omega_{iq} = freq_{iq} \times IDF_i$

$\omega_{id} = freq_{id} \times IDF_i$

$freq_{iq}$ = the frequency of the term i in query Q

$freq_{id}$ = the frequency of the term i in document D

IDF_i = an estimate of the inverse document frequency of term i on the Web

Once the similarity between documents in the base set and the query topic is

established, there are many approaches to deciding if any a document should be eliminated from the set. [5] investigates three approaches based on thresholding the relevance weight: all documents whose weights are below a threshold are pruned as follows:

1. *Median Weight*: The threshold is the median of all the relevance weights.

2. *Start Set Median Weight*: The threshold is the median of the relevance weights of the documents in the root set.

3. *Fraction of Maximum Weight*: The threshold is a fixed fraction of the maximum weight.

Corresponding to the three relevance weight measurements, the authors propose three versions of their improved HITS algorithm: *med*, *startmed*, and *maxby*. Their experiment shows the potential of augmenting the previous pure connectivity analysis-based algorithm with content analysis to make a significant improvement; however, the differences among the three different version are not great.

The *ARC* algorithm of S. Chakrabarti et al. [14] also extends Kleinberg's algorithm with textual analysis. Noting that the text describing the query topic is ignored in the iterative process of the HITS algorithm, they remedy this by altering those sums in Formula (2.2) and (2.3) to be weighted with textual content, so as to maintain focus on the original topic. While in [5], the whole document is used to compute the relevance, in [14], a window surrounding the hyperlink is used to capture the topic of a page. This is based on the authors' fundamental notion: the text around hyperlinks to a page p is descriptive of the content of p. Here, these hyperlinks are not in p, but in pages pointing to p. In particular, if text descriptive of a topic occurs in the text around a hyperlink into p from a good Hub, it reinforces the belief that p is an Authority on the topic. To incorporate this textual conferral of authoritativeness into the basic iterative process of the HITS algorithm, for any page x and y, a $weight(x, y)$ is assigned, if there is a link from page x to page y. As a result,

the Hub and Authority is computed as follows;

$$H_{i+1}(x) = \sum_{(x,s)} W(x,s) A_i(s) \tag{2.4}$$

$$A_{i+1}(x) = \sum_{(p,x)} W(x,p) H_i(p) \tag{2.5}$$

where $W(x, y)$ denotes the the weight between page x and page y, and measures the authoritativeness on the topic invested by page x in y. According to the notion discussed above, if the text in the "vicinity" of the hyperlink from x to y contains text descriptive of the topic at hand, $W(x, y)$ should be increased. There are two questions, however, when applying this notion in practice: (1) what precisely is "vicinity"? and (2) how are the occurrences of descriptive text mapped into a real-value weight? The technique presented in [14] looks on either side of the hyperlink for a window of B bytes, which is called the *anchor window*. That is, B bytes of text between the < a href="..." > and < /a > tags of a hyperlink are included. Suppose there is a hyperlink in page q pointing to page p, and let $n(t)$ denote the number of matches between terms in the topic description and the anchor window of this hyperlink. Then,

$$W(q,p) = 1 + n(t)$$

The authors of [14] also conduct a study to determine B, the parameter governing the width of the anchor window; and the study suggests that most matching occurrences are within 50 bytes of the hyperlinks. Therefore, the parameter B is set to be 50.

In this thesis, the method presented in [5] is adopted to eliminate the possible topic drift problem during the structure analysis in our system The idea of the anchor window in [14] is used to capture the current focused topis of interest to the on-line user.

2.1.5 Web Usage Mining

If users' navigational behavior, such as how and when a user visits a web page, has been tracked and recorded, this usage data would provide us with

an additional information source. In fact, because usage data itself records some users' true behavior, it has the potential to reveal hidden patterns which neither content information nor linkage information are able to, and various research has shown this potential [52] [91] [38] [104]. Usage data were first used for straightforward statistics, such as page access frequency[1]. However, the ability of this approach is obviously very limited. Usage data can actually include more sophisticated forms of analysis, such as finding the common traversal paths through a Web site. In order to do this, some data mining algorithms and techniques are commonly used: among them, the most popular are association rule and clustering techniques. Such analysis of Web usage data is often referred to as **Web Usage Mining**.

Association rule techniques [1] [2] discover unordered correlations between items found in a database of transactions. In the context of Web Usage Mining, a *transaction* is a group of Web page accesses to fulfill one information need, with an item being a single page access. As example of an association rule from the Web site of the University of Alberta Computing Science Department (referred to as the CS Web site below)[2] is the following:
57.81% visitors who accessed the "Program and Admission" page also accessed a page on the graduate program.

The percentage in the example above is referred to as *confidence*. *Confidence* is the number of transactions containing all of the items in a rule, divided by the number of transactions containing the rule antecedents (The rule antecedents in the example are the accesses of the "Program and Admission" page). There is another criteria used in the association rule analysis: *Support*, which is defined as the number of transactions containing all of the items in a rule, divided by the total number of transactions. An association rule captures item dependency in a transaction set, and usually we are interested only in the association rules that have high *support* and *confidence*.

Clustering analysis [35] [64] allows one to group together users or items that have similar characteristics. The clustering criteria is based on domain

[1] http://www.broadvision.com, http://www.webtrends.com, http://www.netgen.com
[2] http://www.cs.ualberta.ca

knowledge and the specific tasks or interests of the analyzers, and the results are often used to perform further processing. An example of a cluster discovered on the CS Web site is the following:
the "Program and Admission" page, the "Graduate Program" page, and the "GAPS" page (which is the graduate application processing system in the department), tend to be accessed together.

As the examples above show, Web Usage Mining techniques have the potential of revealing valuable information. In particular, Web recommender systems which take advantage of usage data have been extensively utilized [92] [25] [49] [100], and we also use such data in building our system. To apply data mining algorithms to usage data, however, there are some requirements. Ideally, the input of the Web Usage Mining process is a collection of user requests that give an exact accounting of who accessed, what Web resources were requested and in what order, and how long each resource was viewed. This format enables us to group users' requests into transactions, each of which is a group of Web resources accessed to fulfill an information need of the visitor. Without accurately identifying these semantically meaningful transactions, data mining algorithms may mistakenly identify incorrect patterns.

However, the usage data collected do not usually fulfill these requirements. In fact, there are a number of difficulties involved in accurately identifying transactions from raw usage data, which make preprocessing one of the major tasks for any Web Usage Mining system, and still an open question. In the following section, we will discuss this issue in detail. Although the usage data can be collected on the client side [109] [86] [106], the majority of Web Usage Mining systems make use of usage data on the server side, where collective information is available. This server side usage data is usually referred as the Web server access log. As a result, Web Usage Mining is also referred to as **Web Log Mining**. In this thesis, we will focus only on the Web server access log.

Term	Description
Remote Host	Remote host name or IP address, from which the request is made
Rfc931	Remote login name of the client
Auth User	Server authenticated client password
Date	Date and time of the request
Offset	Local time offset of the client from Greenwich Time
Method	Method of the request (*GET*, *POST*, etc.)
URL	Full page URL address input by the client
Protocol	Communication protocol used by the client (*HTTP/1.0*, *HTTP/1.1)*
Status	Server status responding to the request and sent to the client
Bytes	Number of bytes transferred to the client in response to his/her request
Referrer	URL that the request originated from
Agent	Name and version of the operating system and browser at the client

Table 2.1: Web Access Log Field Description

Web Server Access Log

The Web server access log is the most commonly used source for Web Usage Mining. Currently, most Web servers provide the option of storing log files in the *Extended Common Log Format (ECLF)*. Table 2.1 lists the fields in this format. In most cases, clients need no user name and password to visit a Web page. Thus, the field *Rfc931* and *Auth User* are empty (indicated by value '-'). The following is an example of an entry in the server access log of the CS Web site using the *ECLF* format:
129.128.4.126 - - [21/May/2004:01:20:06 - 0700] "Get /research HTTP/1.1" 200 8031 /index.html Mozilla/4.0(compatible; MSIE 6.0; Windows NT 5.1)

Please note that a Web server may adopt a different log format, and this could cause a difference in the log preprocessing which follows. In this thesis, we will focus on the most commonly used format (*ECLF*). However, later on we will apply our system to the VIVIDESK system, which provides a very special format of log data. We will discuss the format and its preprocessing work in Chapter 5.

The raw access log recorded in the ECLF format is somewhat arbitrary and incomplete for any data mining algorithm. For example, in most cases, there is no entry to identify users and transactions directly from the log because of the anonymity of Web visits. Worse, due to client and proxy caches, not all client requests are captured in the server access logs, which makes the identification of users and transactions more difficult.

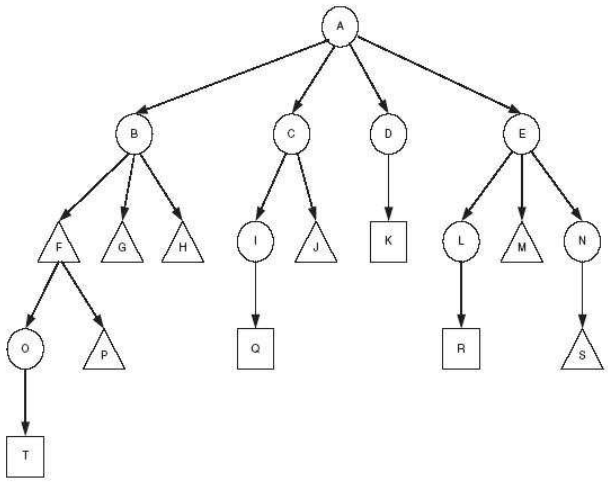

Figure 2.5: A Sample Web Site

In order to group individual web pages recorded in the Web log into meaningful transactions for the discovery of patterns such as association rules, therefore, preprocessing has been performed to reliably identify users and transactions from raw server access logs.

User Identification

In the best case, we can rely on the values in fields *Rfc931* and/or *Auth User* to accurately identify individual users, but in most cases, the two fields are empty. In the absence of such information, some heuristics have to be made to help identify users [21] [77]. If the *Agent* field in the log, which records the name and version of the operating system and browser at the client, shows a change, a reasonable assumption to make is that each different agent type represents a different user. The next heuristic in [21] is to use the access log in conjunction with site topology to construct browsing paths for each user. If a page is requested that is not directly reachable from any visited pages, it is assumeed that there is another user with the same IP address. Consider the Web site shown in Figure 2.5 (the arrows between the pages represent the hyperlink), and a sample server access log shown in Table 2.2 (The first column is for referencing purposes and would not be part of an actual log.) With the two assumptions above, three unique users and their browsing paths are identified:

#	Remote Host	R	A	Date	Method/URL/Protocal	Status	Bytes	Referrer	Agent
1	129.128.4.165	-	-	[25/Apr/2004:03:04:41 -0700]	"GET A.html HTTP/1.1"	200	1152	-	Mozilla/4.0(MSIE 6.0; WinXP)
2	129.128.4.165	-	-	[25/Apr/2004:03:05:34 -0700]	"GET B.html HTTP/1.1"	200	2890	A.html	Mozilla/4.0(MSIE 6.0; WinXP)
3	129.128.4.165	-	-	[25/Apr/2004:03:05:39 -0700]	"GET L.html HTTP/1.1"	200	1300	-	Mozilla/4.0(MSIE 6.0; WinXP)
4	129.128.4.165	-	-	[25/Apr/2004:03:06:02 -0700]	"GET F.html HTTP/1.1"	200	1152	B.html	Mozilla/4.0(MSIE 6.0; WinXP)
5	129.128.4.165	-	-	[25/Apr/2004:03:06:58 -0700]	"GET A.html HTTP/1.1"	200	1841	-	Mozilla/5.0(X11; Linux i686)
6	129.128.4.165	-	-	[25/Apr/2004:03:07:42 -0700]	"GET B.html HTTP/1.1"	200	1210	A.html	Mozilla/5.0(X11; Linux i686)
7	129.128.4.165	-	-	[25/Apr/2004:03:07:48 -0700]	"GET A.html HTTP/1.1"	200	1841	B	Mozilla/5.0(X11; Linux i686)
8	129.128.4.165	-	-	[25/Apr/2004:03:07:55 -0700]	"GET R.html HTTP/1.1"	200	1152	L.html	Mozilla/4.0(MSIE 6.0; WinXP)
9	129.128.4.165	-	-	[25/Apr/2004:03:09:50 -0700]	"GET C.html HTTP/1.1"	200	2762	A.html	Mozilla/5.0(X11; Linux i686)
10	129.128.4.165	-	-	[25/Apr/2004:03:10:02 -0700]	"GET O.html HTTP/1.1"	200	1300	F.html	Mozilla/4.0(MSIE 6.0; WinXP)
11	129.128.4.165	-	-	[25/Apr/2004:03:10:03 -0700]	"GET F.html HTTP/1.1"	200	1152	O.html	Mozilla/4.0(MSIE 6.0; WinXP)
12	129.128.4.165	-	-	[25/Apr/2004:03:10:16 -0700]	"GET B.html HTTP/1.1"	200	1300	F.html	Mozilla/4.0(MSIE 6.0; WinXP)
13	129.128.4.165	-	-	[25/Apr/2004:03:10:45 -0700]	"GET J.html HTTP/1.1"	200	152	C.html	Mozilla/5.0(X11; Linux i686)
14	129.128.4.165	-	-	[25/Apr/2004:03:12:23 -0700]	"GET G.html HTTP/1.1"	200	1680	B.html	Mozilla/4.0(MSIE 6.0; WinXP)
15	129.128.4.165	-	-	[25/Apr/2004:05:05:22 -0700]	"GET A.html HTTP/1.1"	200	2937	-	Mozilla/4.0(MSIE 6.0; WinXP)
16	129.128.4.165	-	-	[25/Apr/2004:05:06:03 -0700]	"GET O.html HTTP/1.1"	200	2050	A.html	Mozilla/4.0(MSIE 6.0; WinXP)

Table 2.2: Sample Web Server Access Log

User 1: A-B-F-O-F-B-G-A-D

User 2: A-B-A-C-J

User 3: L-R

in which the hyphens represent the browsing paths of the users.

Admittedly, these assumptions are only heuristics for identifying users, and cannot guarantee correct identification. Two users with the same IP address that use the same browser on the same type of machine can easily be mistakenly identified as a single user, if they are looking at the same set of pages. Conversely, a single user with two different browsers running, or who types in URLs directly without using a site link structure, can be mistaken for multiple users.

Visit Session Identification

For logs that span long periods of time, it is very likely that users will visit the Web site more than once. The goal of visit session identification, or session identification, is to divide the page accesses of each user into individual sessions, which will become the base on which to identify meaningful transactions later. The simplest method of achieving this is through a timeout where, if the time between page requests exceeds a certain limit, it is assumed that the user is starting a new session. Many commercial products use 30 minutes as a default timeout, and a study [13] established a timeout of 25.5 minutes based on empirical data. As a result, a timeout of 30 minutes is generally used in generic website log data to identify sessions. Using this timeout value, the path for user 1 from the sample log is broken into two separate sessions, since the last two accesses are more than an hour later than the first five. The session identification step results in a total of four visit sessions in the sample log:

Session 1: A-B-F-O-F-B-G

Session 2: A-D

Session 3: A-B-A-C-J

Session 4: L-J

Figure 2.6: Auxiliary-Content Transactions

Transaction Identification

Generally speaking, the reason that users visit a Web site is to search for some information that they need. During each visit session, a user may pursue one, or more than one, information need(s). The goal of transaction identification is to identify, from sessions, pages clicked to fulfill individual information needs during the same visit. If a user pursues only one information need, a whole session can be viewed as a single transaction. If a user has multiple information needs, and each page in his/her visit session fulfills one of them, a session can be viewed as a set of transactions, each consisting of a single page. In most cases, however, a transaction consists of more than one, but not all pages of a session, and the task of transaction identification is to divide and identify such meaningful transactions.

A number of transaction identification approaches have been proposed [21] [17]. R. Cooley et al. identify transactions based on their proposed *auxiliary & content* page model. In this model, they classify Web pages in a Web site into two categories. A *content* page is a page that contains a portion of the information content that the Web site is providing, and which may meet the user's needs; while an *auxiliary* page is simple a page to facilitate the browsing of a user searching for information (to reach content pages). Using this concept, all pages in each session are labeled as either content pages or auxiliary pages. A transaction is defined as a sequence of auxiliary pages (could be zero) that end with a content page, representing a user reaching his/her goal after following a set of auxiliary pages. This is illustrated in Figure 2.6, in which auxiliary and content pages in a visit session are labeled along the time axis with an **A** or **C**, respectively.

There are two underlying assumptions in this transaction identification approach. First, it is assumed that a visitor may have different information

needs to fulfill during a visit, but that all the information needs must be fulfilled in sequence. The second assumption is made in order to identify content pages and auxiliary pages. It is assumed that the amount of time a user spends on a page in a visit correlates to whether the page is an auxiliary or a content page for that user (The time spent on a page is referred to as Reference Length in [21]. Therefore, this approach is often called the **Reference Length** approach). More specifically, it is assumed that a user would spend more time on content pages, and a time has been calculated that estimates the cutoff between auxiliary and content pages. Using a qualitative analysis of several server access logs, [21] gives an estimated cutoff time of 78.4 seconds. According to this estimation, the sessions in the example above are divided into 6 transactions:

Transaction 1: A-B-F

Transaction 2: O-F-B-G

Transaction 3: A-D

Transaction 4: A-B-A-C-J

Transaction 5: L

Transaction 6: R

There are two things worthy of mention in this approach of classifying auxiliary and content pages according to estimated cutoff time. First, the reasonable cutoff time may change for different Web sites. Thus, the estimation needs to be re-done in different contexts. Second, even for the same Web site, the same page may appear more than once in web logs, and it may sometimes be labeled as an auxiliary page, and sometimes as a content page. Therefore, the concept of auxiliary and content pages is applied only at the session level. For this reason, some pages could be considered *multiple purpose* pages from a high-level view. We illustrate this point in Figure 2.5, in which pages classified as auxiliary pages in all sessions in Web logs, are represented using circles; pages classified only as content pages are represented using rectangles; and multiple purpose pages are represented using triangles. This fact reflects, to some extent, the unreasonableness and difficulty of identifying content pages based on stay time alone.

Another transaction identification approach, **Maximal Forward Reference** is proposed, based on the work presented in [17]. In this approach, each transaction is defined to be the set of pages in the browsing path, from the first page in a visit session up to the page before a backward reference is made. A *backward reference* is defined to be a page that is already contained in the set of pages for the current transaction. Similarly, a *forward reference* is defined to be a page not already in the set of pages for the current transaction. A new transaction is started when the next forward reference is made. Again, using the example above, the Maximal Forward Reference approach forms 6 transactions from the sessions:

Transaction 1: A-B-F-O

Transaction 2: A-B-G

Transaction 3: A-D

Transaction 4: A-B

Transaction 5: A-C-J

Transaction 6: L-R

The Maximal Forward Reference approach can be thought of as using the same *auxiliary & content* page model as the Reference Length approach. It also makes the same assumption that a visitor may have different information needs to fulfill during a visit, and that all the needs must be fulfilled in sequence. The Maximal Forward Reference approach is different from the Reference Length approach in the way it classifies auxiliary and content pages in individual sessions. Instead of using time spent on a page, the Maximal Forward Reference approach performs the classification by identifying backward reference points. It has an advantage over Reference Length in that it does not require the input parameter (estimated cutoff time); however, a problem is raised in that a backward reference point does not guarantee that a visitor has found his/her goal. For example, a user could have fulfilled one of his/her information needs in the middle of a navigational path to reach a backward reference point.

To address the identified problems, in this thesis we propose a new user navigation model, and an improved transaction identification approach. In-

stead of identifying transactions based on cutoff time or backward points, we rely on the real content of pages to more accurately identify transactions. We will discuss our approach in detail in Section 3.2.2.

Once transactions have been identified, a Data Mining algorithm can be applied on them to discover patterns. The most commonly used techniques include association rule and clustering, and we will discuss some of these in detail in Section 2.3.

2.2 Web Recommender Systems

Web recommender systems were originally defined as systems in which on-line users provide recommendations as inputs, which the system then aggregates and directs to other appropriate recipients [29]. The term now has a broader connotation, describing any on-line system that produces individualized recommendations as output, or has the effect of guiding the Web user in a personalized way to interesting or useful objects in a large space of possible options [11]. The recommended items could be products, such as books, movies, and music CDs; or on-line resources, such as Web pages or on-line activities [32]. Web recommender systems can recommend Web resources that particularly merit users' attention, saving them needless search and enabling them to reach their goals faster. By recommending, the system can also be used to warn users, for example, that a page is irrelevant to their goals. By doing so, the recommender system could prevent users from losing time during their visits. Web recommender systems, therefore, have the potential to soon become as common as search engines in assisting browsing on the Web [48]. As a matter of fact, some recommender systems have been an integral part of some e-commerce sites, such as Amazon.com[3] and CDNow[4].

A Web recommender system is also referred to as Web recommender agents, Web recommendation system, etc. Generally speaking, a Web recommender system is an interactive software agent, and most such agents work in two

[3]http://www.amazon.com
[4]http://www.cdnow.com

phases: off-line and on-line. During the off-line phase, the agent preprocesses and analyzes available data to build user models, which infer user preferences and interests; in the on-line phase, it uses and updates these models on-the-fly to recognize the user's current goals and predict recommendations.

Recommendation techniques have a number of possible classifications [79] [29]. Recommendations may be anonymous, tagged with the source's identity, or tagged with a pseudonym. Recommender systems can provide advice based upon users' requests, or on their own discretion. The given Recommendations can be presented to users in different ways and interfaces, such as presenting a *Top-N* list, highlighting recommended URLs, or filtering out negative recommendations. For each recommendation candidate, the content of the recommendation can be a single bit (recommended or not), numeric scale (e.g., 1, 2, ..., 5, representing not recommended at all, not recommended, neutral, recommended, and strongly recommended, respectively), or unstructured textual annotations. A Recommender system can also be server-based or client-based. The former resides in a Web server to assist users to navigate within the Web site, while the latter is a program running in users' machines to help them surf through the whole Web. While client-based recommender systems have the merit of being customized for individual users, server-based recommender systems are able to take advantage of collective information.

In this thesis, our interest focuses on the sources of data on which recommendation is based, and the use to which that data is put. Thus, we propose a new classification of Web recommender systems, in terms of the data sources. On this basis, we distinguish three different Web recommender systems: rating-based recommender systems, survey-based recommender systems, and activity-based recommender systems.

2.2.1 Rating-based Recommender Systems

If the recommender system requires web users' explicit participation to judge and rate items encountered so far, before being able to provide any recommendation, it is referred to as a **rating-based recommender system**.

Rating-based recommender systems stem from our social activities. In

everyday life, we often find it necessary to make choices without sufficient personal experience of the alternatives. In this scenario, we rely on recommendations from other people by word of mouth, letters of recommendation, movie and book reviews printed in newspapers, among other sources. The rating-based recommender system assists and augments this natural social process. In a typical rating-based recommender system, on-line users provide recommendations as inputs, which the system then aggregates and directs to appropriate recipients.

One of the most widely implemented, most mature, and therefore most important categories of rating-based recommender systems is the **collaborative filtering recommender system**. In fact, the phrase "collaborative filtering" was coined by D. Goldberg, D. Nichols, and B. M. Oki, the developers of the first Web recommender system [27]. Collaborative filtering systems can produce personal recommendations by computing the similarity between the user's preference and that of other people. The main idea is to automate the social process of "word-of-mouth" by which people recommend products or services to one another. As mentioned above, if an individual needs to choose between a variety of options with which they do not have any experience, they will often rely on the opinions of others who do have such experience. However, when there are thousands or millions of options, as on the Web, it becomes practically impossible for an individual to locate reliable experts who can give advice about each of the options. To make the problem more manageable, collaborative filtering techniques shift from an individual to a collective method of recommendation.

The basic mechanism behind collaborative filtering systems is the following:

- A large group of people are required to register in the system and rate items they have encountered. A profile for each user is built, based on the user's rating;

- Using a similarity metric, the system aggregates the rating objects and recognizes commonalities between users, according to their ratings on items. Users with commonalities are classified into the same sub-groups.

A collective profile is computed to represent the "average" preference of the sub-group;

- For an active user who seeks advice, the sub-group s/he belongs to is identified, and the opinions of people in the same sub-group (whose preferences are similar to the preferences of this user) are used to give recommendation. More specifically, items highly preferred by these people, but not already accessed by the active user, are recommended.

A typical user profile in a collaborative system consists of a vector of items encountered by the user and their ratings, continuously augmented as the user interacts with the system over time. Some systems used time-based discounting of ratings to account for drift in user interests [7] [85]. In some cases, ratings may be binary (like/dislike) or real-valued, indicating degree of preference. The user profile is then grouped based on some similarity metric. Typical similarity metrics are Pearson correlation coefficients between the users' profiles and (less frequently) vector distances or dot products. The similarity metric is also used to identify the preference group.

The greatest strength of collaborative filtering is that this technique can be completely independent of any machine-readable representation of the objects being recommended, and works well for complex objects, such as music and movies, where variations in taste are responsible for much of the variation in preferences. Indeed, if the similarity metric has in fact selected people with similar tastes, the chances are excellent that the options that are highly evaluated by that group will also be appreciated by the advice-seeker. As a result, the typical application of collaborative filtering on a recommender system is the recommendation of books, music CDs, or movies. Of course, this method can be used for the selection of documents, services, or products of any kind. Some of the most important recommender systems using the collaborative filtering technique are GroupLens/NetPerceptions [78] [39] [55], Ringo/Firefly [87], and Tapestry [27]. These systems can be memory-based, comparing users with each other directly using correlation or other measures; however, the more common form is model-based, in which a model is derived

from the historical rating data of users to make predictions [8].

The main bottleneck with existing collaborative filtering systems is the collection of preferences [87]. To be reliable, the system needs a very large number of people (typically thousands) to express their preferences about a relatively large number of options (typically dozens). This requires a considerable effort from a great number of people. Since the system becomes useful only after a "critical mass" of opinions has been collected, people will not be very motivated to express detailed preferences in the beginning stages (e.g., by scoring dozens of music records on a 10 point scale), when the system cannot yet help them. Scalability is another major concern for a practical collaborative filtering recommender system. The matching of the active user profile with each preference group has to be computed as an on-line process. For a practical system with many thousands of users, this may lead to unacceptable latency in providing recommendations.

Another type of rating-based recommender system is the **content filtering system** [43] [70] [31] [4]. As with collaborative filtering, the goal of the content filtering technique is to sort through large volumes of dynamically generated information and present to the user those which are likely to satisfy his or her information requirement. The two systems are different, however, in how they filter out the unnecessary information. In a content filtering system, the objects (e.g., on-line news archives) are represented by their associated features. As in other Web Content Mining systems, most use keywords in documents as representation features. For example, Fab [4] represents documents in terms of the 100 words with the highest TF-IDF weights [81], i.e., the words that occur more frequently in those documents than they do on average. Syskill & Webert system [70] represents documents by the 128 most informative words, i.e., the words that are more associated with one class of documents than another.

During navigation, users are asked to rate the documents they have visited, according to whether and/or how they are interested in the documents. Based on the ratings, a content filtering recommender system compiles a profile of the user's interests. Since all of the documents are represented by their as-

sociated features (e.g., keywords), the profile can also be represented by the designed features. Having derived the user profile, the system makes recommendations based on the content similarity of documents to user profiles. The similarity comparison can be performed using different learning methods, such as Decision Trees, Neural Net, and Bayesian Classifier. For example, Syskill & Webert system [70] employs a Bayesian Classifier to estimate the probability that a document is liked by a user.

Zhu et al.'s work [109] [108] extends the concept of "content filtering" in the recommender system. Rather than relying on a limited range of keywords, their work investigates identifying suitable "browsing features" of words in individual pages to capture and represent users' interests. Recommendations are then given based on this browsing feature matching component of their system.

As indicated above, content filtering recommender systems provide recommendations based on "item-to-item" correlation, while collaborative filtering systems use "people-to-people" correlation to give recommendations [29]. Both, however, require users to rate items they have visited so far, in order to obtain recommendations. Generally, the more user ratings, the better the recommendations the system can provide. This is because in the collaborative case, user profiles are long-term models and are updated as more evidence about user preferences is observed.

2.2.2 Survey-based Recommender Systems

Survey-based recommender systems are defined as systems which ask users to explicitly submit personal information and/or preferences before providing any recommendation. The **demographic recommender system** is the most well-known survey-based recommender system type. It uses demographic information, e.g., personal attributes such as gender, age, or career, to identify the types of users that like certain objects [41] [71]. A demographic recommender system aims to categorize the user based on the demographic information, and make recommendations based on demographic classes. For instance, Lifestyle Finder [41] uses a marketing research survey to gather data regarding

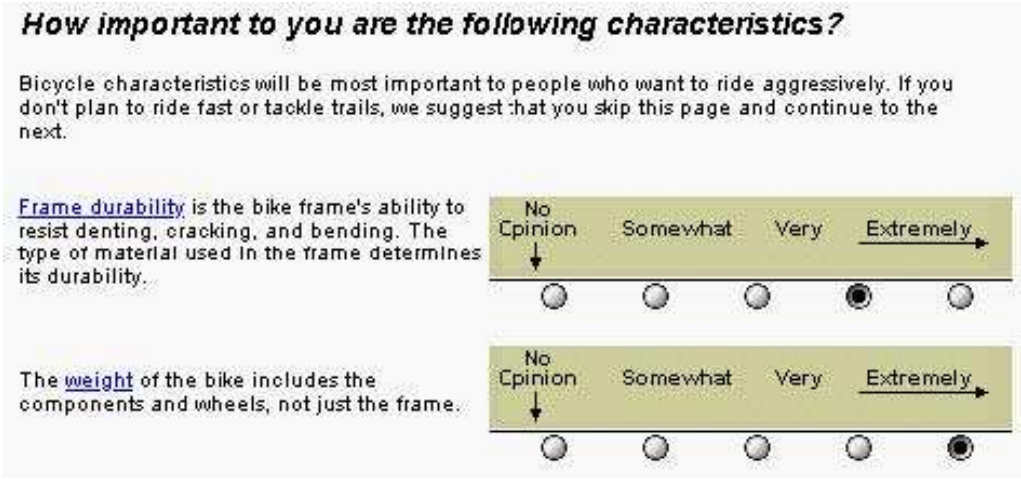

Figure 2.7: Interview Example in the PersonaLogic System [75]

user categorization to suggest a range of products and services. Demographic techniques form "people-to-people" correlation as does collaborative filtering, but they use different data. The benefit of a demographic approach is that it may not require a history of user ratings of the type needed by collaborative and content-based filtering techniques.

There is another type of survey-based recommender system called the **utility-based system** [75] [28], which bases its advice on an evaluation of the match between a user's need and the set of options available. Utility-based recommenders make suggestions based on a computation of the utility of each object for the users. Obviously, the central problem is how to create a utility function for each user; the current systems usually make use of questionnaires to do that. For instance, PersonaLogic [75] helps consumers identify which products best meet their needs by guiding them through a large product feature space in the format of a "deep interview". Figure 2.7 illustrates an example of the questions in the interview.

The system derives the utility function for the users based on the collection of questionnaires, and then employs constraint satisfaction techniques to locate the best match for users, with the assistance of the function. The benefit of utility-based recommendation is that it can factor non-product attributes in the e-commerce environment (e.g., vendor reliability and product availability),

into the utility computation, making it possible, for example, to trade off price against delivery schedule for a user who has an immediate need. However, as we have seen from the example in Figure 2.7, this relies heavily on the information provided by the user.

2.2.3 Activity-based Recommender Systems

In recent years, there has been an increasing interest in building Web recommender systems without input from users. As we can see above, both rating-based systems and survey-based systems rely heavily on users' input – either rating items they have known or answering surveys – to build user profiles, based on which recommendation actions can be determined. The drawbacks of this are as follows:

- explicit user input requires extra work on the part of the user, which is more often than not, either unavailable or considered intrusive;
- the input is often a subjective description of the users by the users themselves, and thus prone to biases;
- the profile is static. It may be good for providing recommendation for a period of time after it is collected; however, its performance degrades over time as the profile ages;
- With the sparsity of such user input, recommendation precision and quality drop significantly.

On the other hand, the footprints users leave while surfing the Web, that is, the activities they performed during their navigation, contain a lot of hidden information concerning the relationship between Web resources and between users. If such user navigation histories are recorded, we can discover hidden knowledge about interesting resources and users without users' explicit ratings or inputs. We call a recommender system which provides recommendations based on users' activity history an **activity-based recommender system**. Such systems have been widely explored [92] [25] [49] [47].

An activity-based recommender system may be located on the client side: Letizia [46] [47] is a typical system in this category. Letizia is a software agent running simultaneously with Netscape when the user surfs the Web, which records and analyzes the user's browsing activity in real time. Over time, Letizia compiles a profile of the user's interests, and presents recommendations based on the user's profile, as well as the current page. Letizia simply uses Netscape as its interface, with one window dedicated to user browsing, and one or more additional windows continuously showing recommendations.

However, the more common and widely-explored activity-based recommender systems are Web usage recommender systems residing on individual Web servers [92] [25] [49] [100], where the collective information is available. Most Web servers have access logs available. As we have shown in Section 2.1.5, the access logs record user browsing and activity history in the server, which contains much hidden information regarding users and their navigation. The access logs could, therefore, be a good alternative to explicit user rating or feedback, in deriving user navigational models for recommendation. Generally speaking, **Web usage recommender systems** take web server access logs as input, and make use of Data Mining techniques, such as association rule and clustering (see Section 2.3) to extract implicit, previously unknown, and potentially useful navigational patterns – which are then used to provide recommendations. A simple example illustrates the basic idea: suppose 90% of former visitors who visited web page A also visited web page B, we can then be confident in recommending page B to a new visitor who is currently focused on page A.

2.2.4 Comparing Recommendation Techniques

We have so far discussed three types of web recommender techniques, each of which has strengths and weaknesses. In this section, we discuss and summarize them, and indicate how they link to the motivation for our work. [11] presents a comprehensive analysis of the advantages and disadvantages of different recommendation techniques.

Collaborative filtering is the most widely implemented and successful tech-

nique, especially for recommending products such as books and movies in e-commerce. However, it suffers from the following problems:

- **New User** Problem: Because recommendations follow from a comparison between the target user and other users, based solely on the accumulation of ratings, a user with few ratings becomes difficult to categorize;
- **New Item** Problem: As with the New User Problem, a new item that has not had many ratings also cannot be easily recommended. This problem shows up in domains such as news articles where there is a constant stream of new items, and each user rates only a few. This is also known as the **Early Rater** Problem, since the first person to rate an item gets little benefit from doing so: such early ratings do not improve a user's ability to match against others. This makes it necessary for recommender systems to provide other incentives to encourage users to provide ratings;
- **Sparse Rating** Problem: Collaborative filtering recommender systems depend on overlap in ratings across users and have difficulty when the space of ratings is sparse. The sparsity of such user input causes the recommendation precision and quality to drop significantly. This is a significant problem in domains such as news filtering, since there are many items available. Unless the user base is very large, the odds that another user will share a large number of rated items is small.
- **Scalability** Problem: In a collaborative filtering system, user profile matching has to be performed on-the-fly as an on-line process. The match usually involves the comparison of very large vectors, which may lead to unacceptable latency for providing recommendations.

These problems suggest that collaborative filtering techniques are best suited to problems where the density of user interest is relatively high across a small and static universe of items. If the set of items changes too rapidly, old ratings will be of little value to new users, who will not be able to have their ratings compared to those of the existing users. If the set of items is large and user

interest thinly spread, then the probability of overlap with other users will be small.

While collaborative filtering systems rely only on user ratings, and recommend items without any textual content or descriptive data, content filtering systems make use of the descriptive data to give "item-to-item" recommendations. Content-based techniques are advantageous in that the goal of people's on-line visits is to find interesting content, and pages related to a given goal in a visit are supposed to be content coherent. As a result, these techniques do not have the New Item problem found in collaborative filtering systems. For example, a TV show recommender can recommend new shows on the basis of their descriptions, even if they have not been rated by anyone. However, these techniques have their own problems:

- **New User** Problem: As with collaborative filtering, content filtering techniques encounter this problem, in that they must accumulate sufficient ratings to build a reliable classifier for users;

- **Document Representation** Problem: Most of the current content-based approaches represent documents by the "important" words or keywords in the documents. However, as we have pointed out, the combination of the keywords in a document is not necessarily able to accurately represent the semantics of the document. The misrepresentation of documents would also create the Fake Similarity problem and the Excessive Similarity problem;

- **Fake Similarity** Problem: Once a representation has been found for documents, a similarity comparison is used to recommend to users the documents which are highly similar to their preferred ones. However, the fact that two documents are similar to each other does not guarantee that people who like one, wish to visit the other at the same time;

- **Excessive Similarity** Problem: In a typical content filtering system, the more similar two documents are, the more likely it is that they will recommend each other. However, people prefer not to read two docu-

ments that contain too much overlapping information. For example, the same news may appear in many newspapers with tiny change in presentation. A news recommender system based purely on content filtering technique might, in fact, annoy readers.

Finding a way to overcome these weaknesses of content filtering techniques while maintaining the benefit of content for recommendation, motivates us to combine content data with usage data to build our hybrid recommender system.

Survey-based recommender systems do not have the New User problem found in the rating-based systems, because they do not require a list of ratings from the user. Instead, they have the task of gathering the requisite information – personal information in demographic systems, for example – which leads to one of the more serious difficulties they have: the **privacy** problem. Asking users to report requisite information may conflict with users' privacy. In fact, the data most predictive of user preference is likely to be information that users are reluctant to disclose. The utility-based systems do not require such personal information. However, users must construct a complete preference function, and must therefore weigh the significance of each possible feature, which often creates a significant burden of interaction. What we would like to build is a recommender system minimizing users' efforts, combined with the greatest endeavor to protect users' privacy.

The root of some of the most severe problems of both rating-based and survey-based recommender systems is that, as their names suggest, the systems heavily rely on users' input to build user profiles and provide suggestions. Activity-based recommender systems, however, have the advantage of being able to give recommendations in the absence of any user input. For instance, Web usage recommender systems take Web server access logs as input, and make use of data mining techniques to extract implicit, previously unknown, and potentially useful navigational patterns, which are then used to provide recommendations. Web server logs record user browsing history, which contains much hidden information regarding users and their navigation. They

could, therefore, be a good alternative to explicit user rating or feedback in deriving user models.

However, a Web usage recommender system which focuses solely on web server access logs has its own problems:

- **Incomplete Information** Problem: One restriction with web server logs is that the information in them is very limited. A number of heuristic assumptions are typically made before applying any data mining algorithm; as a result, some patterns generated may not be proper or even correct.

- **Incorrect Information** Problem: When a web site visitor is lost, the clicks made by this visitor are recorded in the log, and may mislead future recommendations. This becomes more problematic when a web site is badly designed and more people end up visiting unsolicited pages, making them seem popular.

- **Persistence** Problem: When new pages are added to a web site, because they have not yet been visited, the recommender system may not recommend them, even though they could be relevant. Moreover, the more a page is recommended, the more it may be visited, thus making it look popular and boost its candidacy for future recommendation.

The client-side action-based recommender systems have similar problems.

In a summary, all of the recommendation techniques that we have discussed here have their pros and cons, and each would be best suited for some situations. For example, collaborative filtering can be used when the recommended items are products, such as books, movies, and music CDs; while Web usage recommender systems have the power to recommend shortcuts to reach Web pages within a Web site. For each technique, ways to overcome its drawbacks to gain higher performance have been extensively explored. For instance, [24] [80] [83] investigate using a singular value decomposition approach [93] to reduce the dimensionality of the space in which comparison for collaborative filtering takes place. Some researchers attempt to use heuristics

such as time spent reading a page [39], or customers' real purchases [65] to complement users' explicit rating of items. Another method is combine two or more recommendation techniques to gain better performance with fewer of the drawbacks of any individual one. This will be discussed in the following section.

2.2.5 Hybrid Recommender Systems

Hybrid recommender systems are traditionally defined as systems that combine two or more recommendation techniques. There are several ways, however, to do the combining.

Weighted Hybrid Recommender Systems

A **weighted** hybrid recommender system is one in which the score of a recommended item is computed from the results of all of the available recommendation techniques present in the system. For example, the simplest combination would be a linear combination of recommendation scores. The P-Tango system [20] uses such a hybrid. It initially gives collaborative filtering and content filtering recommenders equal weight, but gradually adjusts the weighting as predictions about user ratings are confirmed or discounted.

The benefit of a weighted hybrid is that all of the system's capabilities are brought to bear on the recommendation process in a straightforward way, and it is easy to perform post-hoc credit assignment and adjust the hybrid accordingly. However, the implicit assumption in this technique is that the relative value of the different techniques is more or less uniform across the space of possible items. This may not always be correct (e.g., a collaborative filtering recommender will be weaker for those items with a small number of raters).

Switching Hybrid Recommender Systems

A **switching** hybrid recommender system builds in item-level sensitivity to the hybridization strategy: the system uses some criterion to switch between recommendation techniques. The DailyLearner [6] users a content/collaborative

filtering hybrid in which a content filtering recommendation method is employed first. If the content filtering system cannot make a recommendation with sufficient confidence, then a collaborative filtering recommendation is attempted.

DailyLearner's hybrid has a "fallback" character – one technique is always used first and another technique only comes into play when the former fails. [95] proposed a more straightforward switching hybrid system, in which an agreement between a user's past ratings and the recommendations of each technique are used to select the technique to employ for the next recommendation.

Switching hybrids introduces additional complexity into the recommendation process, since the switching criteria must be determined, and this introduces another level of parameterization. However, the benefit is that the system can be sensitive to the strengths and weaknesses of its constituent recommenders.

Mixed Hybrid Recommender Systems

Where it is practical to make a large number of recommendations simultaneously, it may be possible to use a **mixed** hybrid recommender system, in which recommendations from more than one technique are presented together. The PTV system [89] uses this approach to assemble a recommended program of television viewing. It uses content filtering techniques based on textual descriptions of TV shows and collaborative information about the preferences of other users. Recommendations from the two techniques are combined in the final suggested program.

The PTV case is somewhat unusual because it is using recommendation to assemble a composite entity, the viewing schedule. Because many recommendations are needed to fill out such a schedule, it can afford to use suggestions from as many sources as possible. Where conflicts occur, some type of arbitration between methods is required. In the PTV system, content filtering recommendation takes precedence over collaborative filtering. Usually, however, recommendation requires ranking of items, or selection of a single best

recommendation, at which point the mixed hybrid is not suitable.

Most commonly, hybrid recommendation techniques are investigated to combine collaborative filtering with some other technique in an attempt to avoid the "new user" and "new item" problems. A few hybrid activity-based recommender systems have been proposed in the literature [59] [62]. [59] adopts a weighted hybridization method – more specifically, a clustering technique, to obtain both site usage and site content profiles in the off-line phase. In the on-line phase, a recommendation set is generated by matching the current active session and all usage profiles. Similarly, another recommendation set is generated by matching the current active session and all content profiles. Finally, a set of pages with the maximum recommendation value across the two recommendation sets is presented as the recommendation. [62], on the other hand, utilizes the switching hybridization approach. The authors use association rule mining, sequential pattern mining, and contiguous sequential mining to generate three kinds of navigational patterns in the off-line phase. In the on-line phase, recommendation sets are selected from the different navigational models, based on a localized degree of hyperlink connectivity with respect to a user's current location within the site.

Whether using the weighted method, the switching method, or the mixed method, we find that the process of combination in existing hybrid systems occurs only in the on-line phase. This method suffers from several drawbacks. First, it is only a combination of generated user profiles or models. When generating models, however, there is still only one type of information used: content information, structure information, or usage information. Therefore, these systems do not make full use of all available information to find the useful patterns. Secondly, how to combine the resulting sets to present the most useful recommendations is an open question. In this thesis, we propose a new type of hybrid recommender system. We recognize that there may be distinct information channels available to build users' models for a recommender system. For example, except for Web access log data, the content and structure information of a Web site are possibly available. We could therefore use these data to augment each other in building a better model of users' information

needs. More specifically, our system relies mainly on the Web server access log to build the users' navigational profiles, but also embeds textual content of Web pages and linkage information. For this reason, our system can be viewed as an improved Web usage recommender system. Another distinction of this hybrid system is that the combination could be done in the off-line phase, which can reduce user latency and improve system efficiency. We will talk about our system in detail in Chapter 3.

2.3 Tools for Web Usage Recommender Systems

There are many techniques and algorithms available for Data Mining and Web Mining, and we here present some of those which are related to the building of Web usage recommender systems.

2.3.1 Association Rule Techniques

The **association rule** technique was first introduced in the literature [1] to address market basket data. Let $I = i_1, i_2, ..., i_n$ in which i_k $(1 \leq k \leq n)$ is a binary attribute called *item*. In a transaction database, each transaction T is represented as a set of items such that $T \subseteq I$. Let D be a set of such transactions. Let X be a set of s items such that $X \subseteq I$, called an *s-itemset*. We say that a transaction T contains X, if $X \subseteq T$. An association rule is an expression of the form $X => Y$, where $X \subseteq I$ (rule antecedent), $Y \subseteq I$ (rule consequence), and $X \bigcap Y = \emptyset$.

We define $Support(X => Y)$ as the percentage $s\%$ if $s\%$ of transactions in D contain $X \bigcup Y$, and $Confidence(X => Y)$ as the percentage $c\%$ if $c\%$ of transactions in D that contain X also contain Y. An association rule $X => Y$ holds for D if $Support(X => Y) > minsup$ and $Confidence(X => Y) > minconf$, where $minsup$ and $minconf$ are two threshold values set in the system. *Support* and *confidence* are two important measures of association. We can also represent these two measures as $Support(X \bigcup Y)$ and $Confidence(X \bigcup Y)$. Usually, we are interested in only the association rules

that have high *support* and *confidence*.

An association rule captures item dependency in a transaction set D. Essentially, it answers questions such as: what items do customers often buy together? A classic and interesting association rule found in grocery store data is that people who buy diapers also buy beer.

In general, to find all the association rules $X => Y$ with $Support(X => Y) > minsup$ and $Confidence(X => Y) > minconf$, an association rule algorithm can be divided into two steps. First, all itemsets with *minsup* (called *frequent itemsets*) are generated from the database. This is a computationally expensive step. In the second step, association rules are generated from these frequent itemsets. This step is very straightforward. For a given frequent itemset, $X = i_1, i_2, ..., i_k$, $k \geq 2$, the antecedent of each association rule will be a subset Y of X, and the consequent will be the subset $X - Y$.

Most current algorithms for the discovery of frequent itemsets work iteratively. They first query the database to find all the frequent *1-itemsets*. Then, the frequent *s-itmesets* are generated from the *(s-1)-itemsets* of the previous pass. Since the process of discovering frequent itemsets is expensive over large databases, various techniques have been proposed to speed up the search [2] [84] [69]. Investigators may put constraints on the mining process [97] [72], or prune the extracted rules [51]. Some researchers also report distributed and parallel association rule mining [18] [103].

Association rule techniques are the earliest and most commonly used technique in Web usage recommender systems [92] [25] [49] [15]. The main idea is that an association rule algorithm is picked up and applied on the Web server access logs (after preprocessing), and generates association rules in which *support* and *confidence* are higher than designated parameters, i.e., refer to sets of pages that are accessed together with a *support* value exceeding some specified threshold. These pages may or may not be directly connected to one another via hyperlinks. The antecedents of these rules are used to match the current user's visit page (or a portion of his/her visit history), and if matched, the consequence of the corresponding rule is presented to the user as a recommendation. We have implemented such a recommender system [25] and test our

proposed approach against this technique (see Section 4.3 for experimental results).

2.3.2 Clustering Techniques

The **Clustering** technique is defined as a process of partitioning a set of data (or objects) in a set of meaningful sub-classes, called *clusters*. Each cluster is a collection of data objects that are "similar" to one another and thus can be treated collectively as one group. In order to do so, measurement methods must be employed to measure the similarity between objects, according to some criteria. A good clustering method will produce high quality clusters in which the intra-cluster similarity is high, while the inter-cluster similarity is low. The clustering criteria is based on domain knowledge and the specific tasks or interests of the analyzers, and the results are often used to perform further processing. In the Web Usage Mining domain, there are two kinds of interesting clusters to be discovered: *usage* clusters and *page* clusters. Clustering of users tends to establish 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 provide personalized Web content to the users. The clustering of pages, on the other hand, will discover groups of pages tending to be visited together. This information is especially useful for recommender systems.

User Clustering

User clustering in the context of Web Usage Mining is the process of clustering users based on their behaviors. Sometimes it is impossible to acquire knowledge of a user's interests, but we can guess a user's characteristics based on their behaviors while navigating. Paliouras et al. [66] built user communities by means of clustering. A community corresponds to a group of users who have common interests. In [12], users are partitioned into different clusters by a model-based algorithm. The clustering analysis utilizes a mixture of first-order Markov models using Expectation-Maximization (EM). A first-order Markov model is a Markov model assuming that the probability of occupying a state

at a specified time is determined solely by the preceding state. Each cluster has a different Markov model, and each user belongs to the cluster with some probability. While training, a user is assigned to a particular cluster with some probability, and the initial value of the model is equal. Then, EM is used to determine the proportion of users assigned to each cluster, as well as the parameters of each Markov model.

Page Clustering

Page clustering is the process of grouping pages according to the users' access to them. The intuition is that if the probability of visiting page a, given page b has also been visited, is high, then maybe they can be grouped into one cluster. Page clustering here is very different from text clustering, because for text clustering, the criteria are based on the analysis of the text content, while page clustering is not related to content.

M. Perkowitz and O. Etzioni propose a page clustering method called the *PageGather* algorithm in [74]. PageGather is a type of clustering algorithm which is different from traditional clustering, in that the PageGather algorithm attempts to find a small number of high quality clusters, and these clusters may overlap each other. It is quite reasonable because only subsets of Web pages are of importance to visitors; moreover, there may be multiple themes in a single web page. Another note-worthy point of PageGather is that its goal is to find clusters of related, but not linked, pages. Thus, we adopt this algorithm in our system to cluster Web pages. We will introduce this algorithm in Section 3.2.3.

Mobasher et al. [60] use *Association Rule Hypergraph Partitioning* (ARHP) to perform page clustering. In *ARHP*, the association rule technique is adopted to capture frequent itemsets in the user access dataset. The frequent itemsets are then used to construct a data structure called *hypergraph*. A *hypergraph* is an extension of a graph in the sense that each edge (called *hyperedge*) can connect more than two vertices. In their implementation, each frequent itemset is represented by a hyperedge whose weight is equal to the average confidence of the association rules generated from this frequent itemset. Then, *hMETIS*,

a multi-level hypergraph partitioning algorithm [34], is used to partition the hyperedges, and group pages into individual clusters. hMETIS has been shown to produce high quality bi-sections on a wide range of problems [34], and therefore, is used to create partitions from the frequent itemsets hypergraph.

Chapter 3

A Mission-based Hybrid Web Recommender System

In this thesis, we investigate the design of a hybrid Web recommender system, with three emphases. First, the system is designed to recommend on-line resources (e.g., Web pages) of a specific Web site to its visitors. Second, we attempt to make use of as many information channels as are available to better construct the user profile to be used for recommendation. Third, we take the presence of concurrent information needs of the on-line user into consideration. Pursuing more than one goal, or surfing with concurrent information needs, is fairly common for Web users, but this fact has so far been ignored in Web recommender systems. Our goals are to first accurately identify users' multiple information needs, and then assist them to fulfill their needs by accurately predicting their goals, and recommending Web resources to them. We call the sub-sessions to fulfill individual information needs during an on-line visit *missions*. In our framework, we combine different information channels to identify missions, and then accurately build the user profile based on them. The major information channel we use is the Web server access log, but we also make use of Web content, as well as Web connectivity information. In this chapter, we will discuss the design of this mission-based hybrid Web recommender system, step by step.

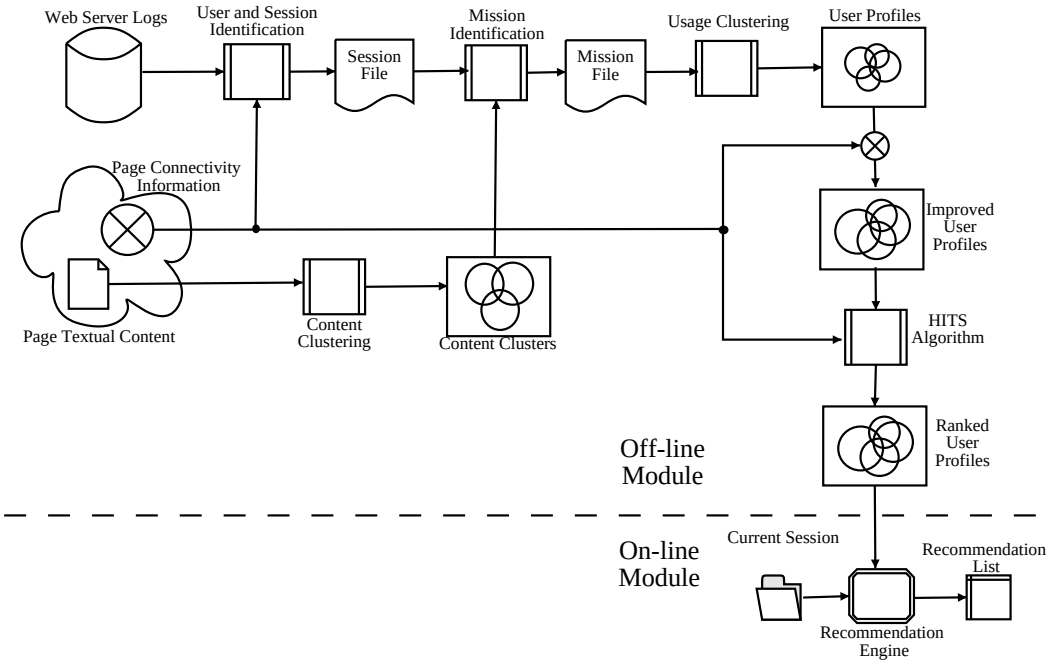

Figure 3.1: System Architecture

3.1 Overall Architecture of the System

As with most Web recommender systems, our system is composed of two modules: an off-line component, which pre-processes data to generate the user profile; and an on-line component, which is a real-time recommendation engine. Figure 3.1 depicts the general architecture of our system.

Generally, entries in the Web server access log are used to identify users and visit sessions, while Web pages or resources in the site are clustered, based on their textual content. These content clusters of Web pages are used to scrutinize the discovered visit sessions in order to identify what we call *missions*. A *mission* is a sub-session with a consistent goal. These *missions* are in turn clustered to generate the user navigational profile. After that, these user profiles are improved with their linked neighbourhood and ranked based on resource connectivity, using the *Hub* and *Authority* ideas [36]. These improved and ranked user profiles are provided to the recommendation engine. When a visitor starts a new session, the session is matched with these clusters to generate a recommendation list. The details of the whole process are given below.

3.2 Off-line Module: Building and Improving the User Profile

To make our system efficient and scalable, the majority of tasks are designed to be done in the system's off-line module. First, Web server access logs are pre-processed so that individual users and visit sessions are identified. Instead of dividing sessions into transactions, as the current activity-based system does, we identify missions from sessions with the assistance of content information. The identified missions are then clustered to discover the user navigational profile to be used for recommendation. Before this profile is provided to the recommendation engine, however, our system also uses Web site linkage information to improve the user profile to better model users' navigational patterns. The structure information is also used to rank the pages in individual clusters.

3.2.1 User and Session Identification

For the typical Web server access log, where each entry contains a client address, the requested data address, a timestamp, and other related information (i.e., the ECLF format as discussed in Section 2.1.5), we use similar pre-processing techniques as in [21] to identify individual users and sessions. The combination of the client's IP address and Agent (including client operating system information and browser information) are used to identify individual users. For sessionizing, we chose an idle time of 30 minutes. Details are provided in Section 2.1.5.

3.2.2 Mission Identification: An Improved Transaction Identification Approach

As discussed in Section 2.1.5, the usual last data pre-processing step of a Web server access log is transaction identification, which divides individual visit sessions into transactions. A number of transaction identification approaches have been proposed. Among them, *Reference Length* [21] and *Maximal Forward Reference* [17] are widely adopted, and accepted as the standards for access data preprocessing. As we have noted in Section 2.1.5, both approaches

share the same underlying assumption: Web pages in each visit session belong to one of two categories: content pages or auxiliary pages; while a transaction is a section of a session, composed of a consecutive sequence of auxiliary pages that ends with a content page. This assumption, however, is problematic in that it requires the sequential ordering in which users' multiple information needs during a visit session are fulfilled. That is, any user must complete one information need before starting another. However, this may not be true in the real Web context. More often than not, we open several browsers to surf a Web site, looking for different information and pursuing several goals at the same time. Moreover, we may sometimes interrupt our current goal and start another in the middle, and then return to the original one later on. In these scenarios, the transaction identification approaches above mistakenly group pages to fulfill users' different information needs into one transaction. Because the transaction is the base of any Data Mining algorithm for pattern discovery, this misclassification would obviously compromise the effect of the Data Mining task, or even cause it to fail. To address this problem, we are here proposing an improved transaction identification approach. To distinguish our approach from the standard approaches, we name the transactions identified using our approach, *missions*. In our mission-based model (in contrast with the transaction-based models), we acknowledge that users may visit a Web site with multiple goals, i.e., different information needs. What's more, we make no assumption on the sequence in which these needs are fulfilled. A mission would model a sub-session related to one of these concurrent information needs, and would allow overlap between missions, which would better represent a concurrent search in the site. While in the transaction-based model, pages are labeled as *content* pages and *auxiliary* pages, and a transaction is simply a sequence of auxiliary pages that ends with a content page, in our mission-based model, the identified sequence is based on the real content of pages. Indeed, a content page in the transaction-based model is identified simply based on the time spent on that page [21], or on backtracking in the visitor's navigation [17]. We argue that missions are better able to model users' navigational behavior than are transactions. For example, a user could

fulfill two goals in a visit session a, b, c, d, in which pages a and c contribute to one goal, while pages b and d contribute to the other. Neither the *Reference Length* [21] or *Maximal Forward Reference* [17] approaches can identify correct transactions in this example, due to their sequential requirement. However, since pages related to a given goal in a visit session are supposed to be content coherent, whether they are neighbouring each other or not, it is possible for the model we propose to accurately identify missions, using real page content.

More specifically, all Web site pages are clustered based on their content in our system, and these clusters are used to identify content coherent clicks in a session into individual missions. Let us give an example to illustrate this point. Suppose the text clustering algorithm groups web pages a, b, c, and e; web pages a, b, c, and f; and web pages a, c, and d into three different content clusters (please note that our text clustering algorithm is a soft clustering one, which allows a web page to be clustered into several clusters). Then, for a visit session a, b, c, d, e, f, our system identifies three missions as follows:

Mission 1: a, b, c, e

Mission 2: a, b, c, f

Mission 3: a, c, d

As seen in this example, mission identification in our system is different from transaction identification, in that we are able to group web pages into one mission, even if they are not sequential in a visit session. However, we can see that our mission-based model generates the transaction-based model, since missions could become transactions if visitors fulfill only one information need in a visit, or fulfill their information needs sequentially. For instance, using the same visit session as in the last example, suppose the user has two information needs, and s/he fulfills them sequentially by visiting a, b, c, and d, e, f, respectively. In this situation, it is very likely that Web pages a, b, and c and Web pages d, e, and f would be grouped into two content clusters. As a result, both our mission-based approach and the previous transaction identification approaches would identify two missions (transactions):

Mission/Transaction 1: a, b, c

Mission/Transaction 2: d, e, f

From the same example above, we can see another difference between our mission identification and the traditional transaction identification. Our adoption of soft clustering makes it possible that a page could participate in more than one mission, while each page belongs to one and only one traditional transaction. This better models the fact that Web pages may be multi-purposed, that is, the same page could be helpful for fulfilling users' different information needs.

To cluster Web pages based on their content, we propose a content clustering algorithm called $DC-tree^{+}$. This is a modified version of the *DC-tree* algorithm proposed in [98]. We selected the DC-tree algorithm because it is specifically designed to classify Web documents. More importantly, the DC-tree algorithm does not require the number of clusters to be discovered as a constraint, but allows the designation of the preferred cluster size. This was the appealing property to us. Indeed, we do not want either too large or too small content cluster sizes. Very large clusters cannot help capture missions from sessions, while very small clusters may break potentially useful interrelations between pages in terms of access usage in sessions.

The original DC-tree algorithm, however, was a hard clustering approach, prohibiting overlap of clusters. We modified it to allow Web pages to belong to different clusters in our DC-tree^{+} algorithm. In both the original and our modified DC-tree algorithm, each Web page is represented as a feature vector; a feature is represented by a keyword extracted from Web pages. After that, Web pages are organized in, and clustered by, a tree structure called *DC-tree*. In the following sections, we present the DC-tree algorithm, as well as the modifications made to it, in order to make it a soft clustering approach (i.e., allowing cluster overlap).

Web Page Feature Extraction

The first sub-task in the DC-tree algorithm is to represent each Web page by a feature vector. Originally designed to automatically cluster Web pages returned from search engines to narrow down the search scope, a novice feature extraction approach is proposed in the DC-tree algorithm, especially designed

Algorithm Feature Extraction in DC-tree Algorithm
Input: A collection of Web documents
Output: A set of keywords
Procedure
1. Randomly select a subset of documents with size m from the collection.
2. Extract the set of words that appear at least once in the documents. Remove stop words and combine the words with the same root by using the stemming technique.
3. Count the document frequency of the words which are extracted in Step 2.
4. Set $lower = k$ and $upper = k$
5. Select all words with document frequency in the range from *lower* to *upper*
6. Check if the *coverage* of these words is larger than the pre-defined threshold. If so, stop. Otherwise, set $lower = lower - 1$ and $upper = upper + 1$, and then goto Step 5.

Figure 3.2: Feature Extraction Sub-system in DC-tree Algorithm

to prevent either very large or very small clusters. Very large clusters cannot help to narrow the search scope, while very small clusters can increase the total number of clusters, and may actually be caused by noise. In order to do this, a parameter k is used to set an *approximate* number on the cluster size when extracting features, representing that the resulting cluster to contain about k Web pages are preferred. Hence, the number of clusters is approximately N/k, where N is the total number of web pages to be clustered. The feature extraction sub-system in the DC-tree algorithm is shown in Figure 3.2.

In Figure 3.2, in order to extract the representative features from the Web page collection efficiently, the algorithm randomly selects a set of sample pages for feature extraction, in Step 1. Since shorter feature vectors lead to shorter clustering time, Steps 4 to 6 try to minimize the number of features and obtain reasonable coverage for them. The *coverage* of the features is defined as the percentage of documents containing at least one of the features extracted. Assume the user wants the resulting cluster to contain k documents. In the ideal case, a feature for a cluster will appear only in the cluster and, hence, the document frequency of the feature is k. Therefore, the algorithm first selects

the features with document frequency equal to k, by setting *lower* and *upper* to k in Step 4. The range {*lower, upper*} is enlarged repeatedly in Step 6 to ensure sufficient coverage for the resulting feature set; this is done by applying a pre-defined *coverage threshold*. Here, we can also see that N/k is only a rough guideline; the actual number of clusters of the clustering result may not be the same as N/k. In the experiments in [98], 0.8 is given as a good coverage threshold value. After extracting document features, each Web page P_i is represented in the following form: $P_i = (ID_i, W_i)$, where ID_i is the page identifier which can be used to retrieve page P_i, and W_i is the feature vector of page P_i: $W_i = (w_{i1}, w_{i2}, ..., w_{in})$. Here, n is the number of extracted features, and w_{ij} is the weight of the j-th feature, where $j \in \{1, 2, ..., n\}$. w_{ij} is equal to 1 if P_i contains the j-th feature. Otherwise, w_{ij} is equal to 0.

Document Cluster (DC) and DC-tree

A *Document Cluster* (DC) is a data structure for storing the information about a set of documents to be grouped into the same cluster. Given N documents in a cluster: $D_1, D_2, ..., D_N$, the Document Cluster (DC) of the cluster is defined as a triple: $DC = (N, ID, W)$, where N is the number of documents in the cluster; ID is the set of the document identifiers of the documents in the cluster, i.e, $ID = \{ID_1, ID_2, ..., ID_N\}$; and W is the feature vector of the cluster, i.e, $W = (w_1, w_2, ..., w_n)$, where $w_j = \sum_{i=1}^{N} w_{ij}$, and n is the number of extracted features.

This triple becomes the component data structure of the DC-tree algorithm. This algorithm organizes Web pages into a tree structure called *DC-tree*, in which each non-leaf node can be considered as a Document Cluster. The algorithm scans the collection of Web pages, guiding each incoming document to an appropriate Document Cluster at the leaf nodes. This structure is similar to the B^+-tree, and is designed so that assigning a document to a cluster requires visiting only a small number of nodes. In detail, for each incoming document D, represented by a feature vector *V(D)*, the DC-tree algorithm recursively descends the DC-tree, starting from the root, to search the *close* child DC node of D. The closeness is measured with some similarity value between

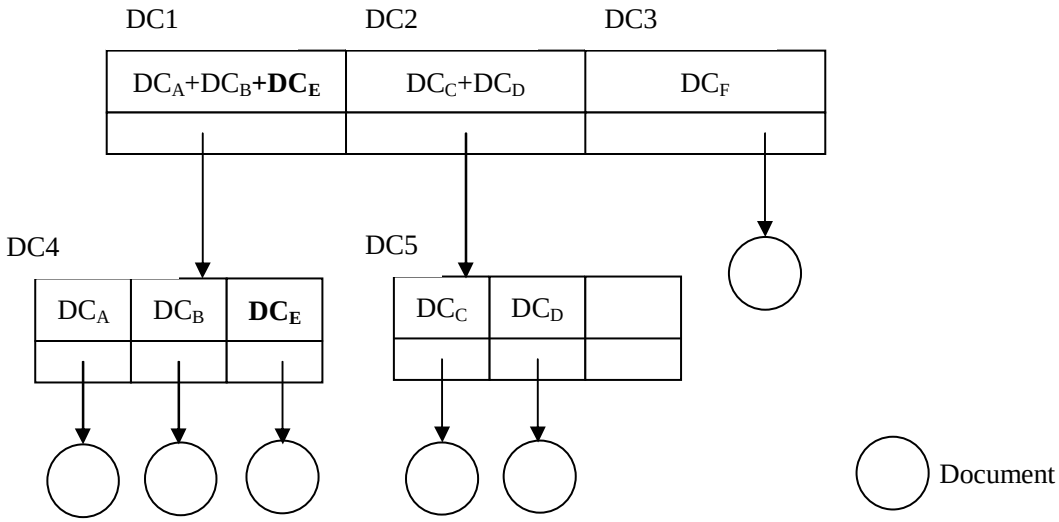

Figure 3.3: Example of a DC-tree Insert (A): Original DC-tree Algorithm

V(D) and the feature vector of the individual DCs. In [98], three alternatives for measuring the similarity – *Cosine Correlation*, *Euclidean Distance*, and *Manhattan Distance* – were tested. The experiment in [98] shows that *Manhattan Distance* always gives the best performance. Therefore, we chose to use Manhattan Distance in our implementation. The Manhattan Distance similarity between two items represented by feature vectors, $W_1 = (w_{11}, w_{12}, ..., w_{1n})$ and $W_2 = (w_{21}, w_{22}, ..., w_{2n})$, is defined as:

$$Similarity(W_1, W_2) = 1 - \frac{\sum_{i=1}^{n} |\frac{w_{1i}}{n} - \frac{w_{2i}}{n}|}{n}$$

If such a close child DC node is found, the document D is inserted into it (e.g., the document E is inserted into DC4 in Figure 3.3). If such a child node does not exist, D is inserted as a new leaf node (e.g., the document F in Figure 3.3). After the document is inserted, all non-leaf entries on the path from the root must be updated to reflect the addition of this new document. The process of inserting document E is shown in Figure 3.3, as an example. Before the insert, $DC_1 = (2, (ID_A, ID_B), (w_{A1}+w_{B1}, w_{A2}+w_{B2}, ..., w_{An}+w_{Bn})$, where w_{Ai} is the value of the i-th feature of document A, while w_{Bi} is the value of the *i-th* feature of document B. After the insert, the Document Cluster value will be updated as $DC'_1 = (3, (ID_A, ID_B, ID_E), (w_{A1}+w_{B1}+w_{E1}, w_{A2}+w_{B2}+ w_{E2}, ..., w_{An} + w_{Bn} + w_{En})$.

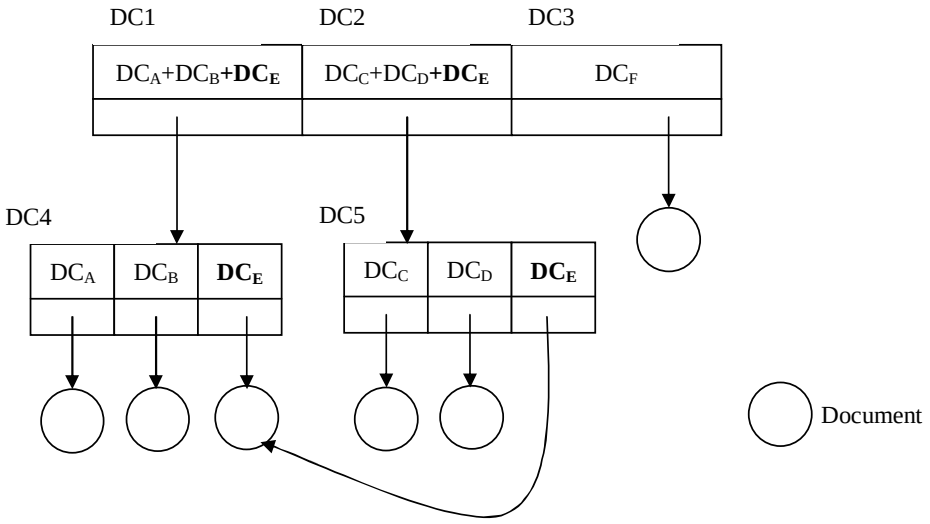

Figure 3.4: Example of a DC-tree Insert (B): Our $DC-tree^{+}$ Algorithm

In the original DC-tree algorithm, any document is inserted into the *closest* child node with the highest similarity value. For instance, in the example illustrated in Figure 3.3, Document E is inserted only into the cluster represented by DC1. This makes it a hard clustering algorithm; that is, a document cannot belong to more than one cluster at the same time. Recognizing that a Web page may be designed to contain information for multiple topics, we prefer to allow Web pages to belong to different clusters. Thus, we modify the DC-tree algorithm so that a document will be inserted into any close child nodes with a similarity value higher than a threshold S. In the example above, suppose the document E is similar to both cluster DC1 and cluster DC2, then our algorithm inserts the document into both of them, as illustrated by the insert of document E in Figure 3.4.

For practical implementation, there is another parameter for the DC-tree algorithm: *Branching Factor* (B), which is used to limit each node to contain at most, B entries. We therefore must deal with the case in which there is no available space for inserting. In that scenario, node splitting must be performed, in order to add a new entry to a DC node containing B entries. In detail, when inserting the *B+1* entry, we partition the set of *B+1* entries into two groups, one for each new node. The division is done in such a way that the similarity between the two new nodes will be minimized and the similarity

among the documents within the same node will be maximized [98].

Identifying the Interesting Clusters

After inserting and organizing Web pages in a DC-tree, a breadth-first search algorithm is applied to discover the clusters that we are interested in. An interesting cluster is defined as a cluster that fulfills the following two conditions:

1. All of the Web pages in the cluster are coherent to a topic;

2. The size of the cluster is in a predetermined range.

Condition (1) has been fulfilled in the DC-tree algorithm by inserting and organizing Web pages into the DC-tree. We now make use of the *lower* and *upper* values found in the feature extraction method (see Section 3.2.2) to determine the cluster size range in Condition (2). Let l and u be the lower bound and upper bound of the cluster size range, then l and u can be determined by the following equations: (a) $l = lower \times N/m$, (b) $u = upper \times N/m$, where N is the data set size, and m is the size of the sample data set used in the feature extraction. Once we have determined the cluster size range, we choose all DC nodes with Web pages within l and u as interesting clusters. Please note that the information of pages in the same cluster has been summarized by a corresponding DC node. Thus, once we have identified an interesting cluster, the cluster can be represented by the feature vector of the corresponding DC node.

The parameters we adopt to run the DC-tree algorithm in our experiment are as follows: approximate number of the cluster size $k = 20$ (the resulting lower bound and upper bound are 2 and 39, respectively); the size of the subset for feature extraction $m = 6000$ (about 20% of the number of total Web pages); branching factor $B = 20$; similarity threshold $S = 0.3$. These parameters have been chosen by trial and error, meaning that we have tried different combinations of values and discovered that the chosen values give comparably good results.

The content clusters derived after running the modified DC-tree algorithm are used to divide sessions into missions. There is one more thing that needs to be pointed out in the mission identification process. Our DC-tree algorithm cannot guarantee each Web page will be grouped in one of the resulting content clusters. As a result, there may be some clicks in a session which cannot be classified into any mission, based on the content clusters. For the purpose of keeping the interrelationship between pages in terms of usage, we attach these pages into all the identified mission(s) from the same session. For instance, in the example we give in the earlier part of this section, suppose there is one more page g, which cannot be clustered in any of the three content clusters. Then, page g is attached to all three missions as follows:

Mission 1: a, b, c, e, g

Mission 2: a, b, c, f, g

Mission 3: a, c, d, g

3.2.3 Clustering the Missions: Building User Profiles

After identifying the missions, we discover patterns regarding users' navigation in the Web site from the mission file, which may be used by the recommender system to make recommendations for users with similar navigational behavior. The action of this pattern discovery can also be viewed as a branch of user profiling/modeling in the field of Information Retrieval, which is generally defined as the process of gathering, organizing, and discovering information to create the summarization or description of the user and/or user interests [68].

For the Web recommender system, user profiling can be done on two levels: personal level and mass level. In the former, the user profile is built for individual users; while in the latter, the profile is built based on a mass of users, as a whole. Most rating-based recommender systems (Section 2.2.1) and survey-based recommender systems (Section 2.2.2) are examples of personal user profiling. This type of system allows fine-grained recommendation, since the recommendation is customized for each user. However, as we have pointed out, the system relies heavily on explicit user input, which may be either unavailable or considered intrusive. On the other hand, in the activity-based

recommender system, without individual user information, the user profile can be built only on the mass level, which is actually an overall navigational preference for the general user.

To build the (mass) user profile in our system, we simply group the missions we uncovered into clusters of Web pages. This process is similar to other Web Usage Mining systems [58] [63] [99] [86], in which each cluster groups a set of Web pages that are frequently visited together, representing a user profile in terms of navigational behavior. However, the user profile discovered from the mission has its own distinction: each cluster in our system is a set of Web pages that are not only frequently visited together, but also have coherent content. In other words, the user profiles discovered from missions possess two characteristics: usage cohesive and content coherent. Usage cohesiveness means the pages in a cluster tend to be visited together, while content coherence indicates that pages in a cluster tend to be related to a topic or concept (This is because missions are grouped according to content information). Since each cluster is related to a topic, and each page has been represented in a keyword vector, we are able to easily compute the topic vector of each cluster. In fact, this topic vector has been represented in the corresponding DC node in the DC-tree (see Section 3.2.2). The cluster topic is widely used in our system, in both the off-line and on-line phases (see below for details).

The clustering algorithm we adopted for grouping missions is *PageGather* [73]. This algorithm is a soft clustering approach, allowing overlap of clusters. Instead of attempting to partition the entire space of items, it attempts to identify a small number of high quality clusters, based on the *clique* clustering technique. This property makes it suitable for a recommender system. The algorithm is briefly described in Figure 3.5 [73].

Here we discuss each step in Figure 3.5, in turn.

1. **Compute the co-occurrence frequencies between pages**. For each pair of Web pages, *P1* and *P2*, in the mission file, we compute $P(P1|P2)$, the probability of a visitor visiting *P1* if s/he has already visited *P2*; and $P(P2|P1)$, the probability of a visitor visiting *P2* if s/he has already

Algorithm PageGather
Input: Missions
Output: A set of page clusters (*cliques*)
Procedure PageGather
1. Compute the *co-occurrence frequencies* between pages
2. Create the *similarity matrix*
3. Create the graph corresponding to the matrix, and find *cliques* in the graph
4. Return the found *cliques*

Figure 3.5: PageGather Algorithm

visited *P1*. We call the minimum of the two values the *co-occurrence frequency* between *P1* and *P2*.

2. **Create a similarity matrix**. We create a matrix called the *similarity matrix*. We set the matrix cell for two pages to the co-occurrence frequency between the two pages if there are no links between them; or zero, if they are already linked in the site. In order to reduce noise, we apply a threshold to remove edges corresponding to low co-occurrence frequency.

3. **Create the graph corresponding to the matrix**. We create a graph in which each page is a node and each nonzero cell in the matrix is an arc. In this graph, a cluster corresponds to a set of nodes whose members are directly connected with arcs. A *clique* – a subgraph in which every pair of nodes has an edge between them – is a cluster in which every pair of pages co-occurs often.

4. **Return found cliques as the found clusters**.

3.2.4 Augmenting and Pruning the Clusters: Improving User Profiles

The missions we extracted and clustered to generate the user profile are based primarily on the sessions from the Web server access logs. These sessions exclusively represent Web pages or resources that were visited. It is conceiv-

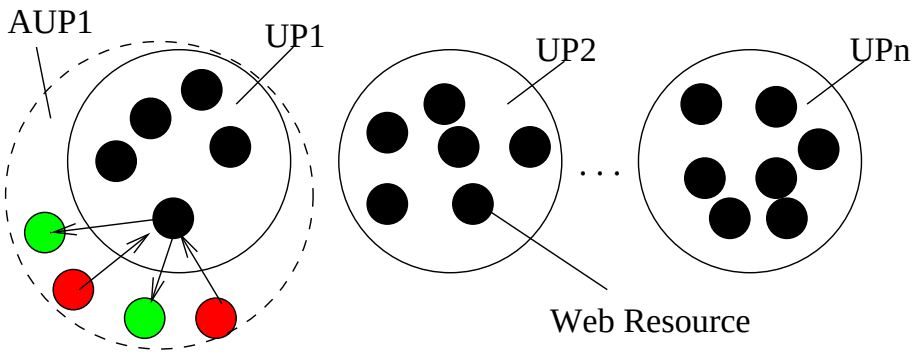

Figure 3.6: User Profiles(UPs) and Augmented User Profiles(AUPs)

able that there are other resources not yet visited, even though they may be relevant, interesting, and worth including in the recommendation list. Such resources could be, for instance, newly added Web pages, or pages that have links to them not evidently presented due to bad design. Thus, these pages or resources are never presented in the missions previously discovered. Since the user profile, represented by the clusters of pages in the missions, is used by the recommendation engine, we need to provide an opportunity for these rarely visited or newly added pages to be included in the clusters; otherwise, they would never be recommended. To alleviate this problem, we expand our clusters to include the connected neighbourhood of every page in a mission cluster. The neighbourhood of a page p is a set of pages directly linked from p, and pages that directly link to p. Figure 3.6 illustrates the concept of neighbourhood augment. This process also helps to avoid overfitting, i.e., avoids recommending only what has been visited.

In more detail, this approach of expanding the neighbourhood is performed as follows: we consider each previously discovered user profile (i.e., a cluster of content coherent and visitation cohesive missions) as a set of seeds. Each seed is supplemented with pages it links to, and pages from the Web site that link to it. The result is what we is called a connectivity graph which now represents our augmented user profile. This process of obtaining the connectivity graph is similar to the process used by the HITS algorithm [36] to find the *Authority* and *Hub* pages for a given topic (see Section 2.1.4). The difference is that we do not consider a given topic, but start from a mission

cluster as our set of seeds. Please note, however, that each cluster obtained in our system so far has been content coherent, that is, consists of Web pages with a specific topic. That is the rationale for treating each cluster as a set of seeds comparable to a root set in the HITS algorithm. In addition, we consider only internal links, i.e., links within the same Web site. By extending the clusters with connectivity information, the augmented user profile includes a local neighbourhood of pages of the original clusters, which could include Web pages that have not appeared in access logs. Because two pages are linked only when the authors thought that a user who was viewing one might want to view the other, the connection is a good indication of relevance. Therefore, the extended cluster is a good approximation of the "semantic neighbourhood" of the original clusters, which could contain important pages in terms of the topic of the original user profile. Thus, this augmented user profile could recommend to users "important" but "unexpected" resources, e.g., new added authoritative pages. However, not all the pages in the local neighbourhood of a cluster are coherent in the same topic of the cluster. As a result, including all pages in the local neighbourhood may compromise one of the benefits of the user profile in our system: content coherence. To eliminate this problem, a further step is included in order to continue to improve the user profile. In detail, we compute content relevance weights of all supplemented pages in each augmented cluster. The content relevance weight of a page equals the similarity of the page content to the corresponding mission cluster, which is represented by the cosine normalization of Web pages and mission clusters' keyword vectors. We then prune nodes, whose relevance weights are below a threshold, from the connectivity graph. This process is similar to the one proposed in [5] that aims at eliminating the "topic drift" problem of the original HITS algorithm (see Section 2.1.4). According to the experimental comparison in [5], we use *Median Weight* (i.e., the median of all relevance weights) as the pruning threshold in our system. The pruning process avoids augmenting the user profile with pages that focus on other topics, and guarantees that the augmented profiles are still topic coherent and focused. After expanding and pruning the clusters representing the user profile, we also augment the feature

vectors that label the clusters. The new keyword vectors that represent the improved user profile also have the keywords extracted from the content of the augmented pages.

In addition, we made one attempt at performing the expansion step to include neighboring pages twice, thus including pages (with content pruning) which have a link-distance two or less from at least one page in each original user profile. We theorized this might include more "important" but "unexpected" resources as desirable, but experiments show that its performance is worse than doing the expansion only once. Thus, we continue to perform the expansion once, as the original HITS algorithm does.

The structure information is also used in our system to rank the pages within the cluster. We recognize that determining how to present the candidate recommendation items in a suitable order is an important issue in building a Web recommender system, especially when the number of recommendation candidates is large. For example, suppose a user is on Web page a, and the system finds 10 other pages the user may be interested in. Which ones should be presented as the top 3 of the recommendation list? Which one should be selected if only a single recommendation can be given? If a recommender cannot rank and present its recommendation items in a proper order, users would lose trust in it, even though its recommendation list includes items that are actually desirable. Although a large number of Web recommender systems have been proposed, few systems have so far dealt with this important issue. Most systems simply provide recommendation lists, treating all items in a list as equal.

We take advantage of the constructed connectivity graph of clusters and, apply the HITS algorithm on them to identify the Authority and Hub pages within each of them. These measures of Authority and Hub allow us to rank the pages within the cluster, and this ranking will be used for recommendation presentation order later in the on-line module.

As mentioned in 2.1.4, Authority and Hub are mutually reinforcing concepts [36]. Indeed, a good Authority is a page pointed to by many good Hub pages, and a good Hub is a page that points to many good Authority pages.

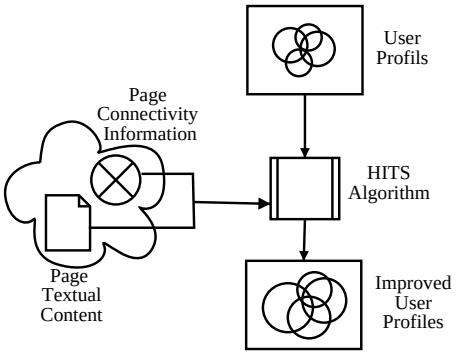

Figure 3.7: User Profile Improvement Process

Since, in our framework, we would like to be able to recommend pages newly added to the site, we consider only the Hub measure. This is because a newly added page would be unlikely to be a good authoritative page, since not many pages are linked to it. However, a good new page would probably link to many Authority pages; it would, therefore, have the potential to be a good Hub page. Consequently, we use the Hub value to rank the candidate recommendation pages in the on-line module.

In sum, our system makes use of structure and content information to improve the user profiles built from missions. Connection information is also used to rank pages within the improved user profiles. The process is summarized in Figure 3.7. Both the methodologies of constructing the user profile in multiple steps (building and then improving), and ranking items in the profile with Web structure information, are new in the field of Web recommender systems.

3.3 The On-line Module: The Recommendation Engine

The previously described process consists of work done exclusively off-line. The on-line module in our framework, on the other hand, is a recommendation engine which gives recommendations to visitors of the Web site on-the-fly. Generally speaking, the recommendation engine in a recommender system is triggered to give recommendations with some trigger mechanism. The trigger

could be something bought, a special date, etc. In our framework, the trigger is the user's current information need. When a visitor starts a new visit on the Web site, we identify his/her information need, and try to match it on-the-fly with already captured user profiles. If they match, we recommend the most relevant pages in the matched cluster. Please note that due to the concern for user privacy, we delete the behavior record of the current user as soon as s/he leaves. If the user returns, new recommendation are provided, based on his or her new behavior.

Identifying the information need of the current visitor consists of recognizing the current focused topic of interest to the user. A study in [14] shows that looking on either side of an anchor (i.e., text encapsulated in a *href* tag) for a window of 50 bytes would capture the topic of the linked pages (see Section 2.1.4). Based on this study, we consider the anchor clicked by the current user, and its neighbourhood on either side, as the contextual topic of interest. The captured topics are also represented by a keyword vector, which is matched with the keyword vectors of the clusters representing the improved user profile. From the best match, we get the pages with the best Hub values and provide them in a recommendation list, ranked by the Hub values. To avoid supplying a very large list of recommendations, the number of recommendations is adjusted to make it proportional to the number of links in the current page. Our goal is to have a different recommendation strategy for different pages, based on how many links the page already contains. Our general strategy is to give $\sqrt{n}$ best recommendations (n is the number of links), with a maximum of 10. This strategy is applied in order to prevent adding noise and providing too many options. The relevance and importance of recommendations is measured with the Hub value already computed off-line.

Chapter 4

Evaluation of the Recommender System

In Chapter 3, we discussed how we utilize the usage, content, and structure data of a Web site to build the user profile, and use it to make recommendations. The immediate questions, then, would be: 1) what is the performance of our recommender system, especially when compared with other similar ones? And 2) to what extent do the usage, content, and structure information actually contribute to the improvement in recommendations?

Thus, in this chapter we talk about the evaluation of our recommendation system. In fact, with Web recommender systems becoming more widely used, a careful evaluation of their performance becomes increasingly important. However, recommender systems are complex applications that are based on complicated models, and this complexity makes evaluation efforts very difficult. In addition, recommender systems are designed for different contexts. Choosing proper methodologies and metrics for evaluation of the diverse systems is challenging, and results are hardly generalizable. As a consequence, many recommender systems in the literature ignore this evaluation and comparison work. As an alternative, they seek some examples to show the "rationality" of the results their systems generate. In this thesis, however, we explore how to choose a proper evaluation methodology, as well as suitable mathematical metrics to evaluate the overall performance of our hybrid recommender system. Moreover, the methodology and metrics are used to assist us in evaluating the impact of the different information channels we use in

improving recommendations.

From our point of view, the quality of a Web recommender system should be measured by its *efficiency* and its *effectiveness*. While *efficiency* can be easily measured by the latency between the user's request and the system response in presenting the recommendation, *effectiveness* can be measured in two ways: *transparency* and *trust*. Transparency means that the system can explain the recommendation it gives. That is, the system gives the recommendation when it has confidence and can satisfy a user's challenge, such as "why do you recommend this to me?". At other times, the system seems transparent to users without any intrusion on their surfing. Trust measures whether and how the user likes the recommendation, which would contribute and add to the user's trust in the system. This can be compared to work done with the assistance of a secretary. A competent assistant has a good idea of what you might like, as well as knowing when s/he should appear. We recognize that the user's trust in a recommender system stems from the user profile the system uses for recommendation.

In the following sections, we first identify possible evaluation methodologies and commonly used metrics to evaluate the recommender system. After that, we describe our evaluation methodology and metrics, and apply them to the University of Alberta Computing Science Department Web site data, to test and evaluate our system.

4.1 Evaluation Methodologies Overview

Let us review briefly how an activity-based Web recommender system is constructed. After preprocessing the data collection, a Data Mining algorithm (e.g., a clustering algorithm in our system) is applied to the data to generate the user profile. The recommender system utilizes the user profile in the on-line module to give the recommendation to users, on-the-fly. We recognize from this process that three evaluation methodologies could be used to evaluate a Web recommender system.

4.1.1 Evaluation of the Algorithm

The first method is to evaluate the Data Mining algorithm the recommender system adopts. For example, some recommender systems use clustering technology to group either users who have common interests, or Web pages that are frequently visited together. The recommender system then takes advantage of the resulting clusters to generate recommendations. Because the clustering algorithm is the core and most important component in building such a system, we may evaluate the performance of the recommender system by evaluating the quality of the clusters. For two recommender systems, each of which uses a different clustering algorithm to generate clusters, the system whose clustering algorithm generates higher-quality clusters can be argued to be better than the other. However, there are several major shortcomings with this methodology:

- This approach emphasizes how well patterns in the data set can be learned, rather than how useful the patterns are for recommendation purposes. The quality of a clustering algorithm cannot be simply equal to the quality of a recommender system based on it. The clusters generated by a clustering algorithm may be very helpful for some applications, but it is not certain that a recommender system based on them is the most optimal;

- A Web recommender system is a complicated system, but this approach ignores many aspects in the system which could contribute to system performance. For instance, as we have pointed out, the problem of how to present the candidate recommendation items in a suitable order is an important issue in building the Web recommender system. Evaluation of the algorithm cannot identify any endeavor of improving recommendation quality by adjusting recommendation order.

- It is impossible to compare different types of recommender systems using this methodology. One of the main purposes for evaluating recommender systems is to compare different recommenders. We cannot achieve this purpose if we only evaluate the algorithms they use. For example, sup-

pose there are two recommender systems – one based on the association rule algorithm and the other based on the clustering algorithm. The result of the association rule algorithm is a set of frequent itemsets, while the result of the clustering algorithm is a set of clusters. It is both meaningless and virtually impossible to compare a frequent itemset and a cluster set, and to say which one is better.

4.1.2 User-centred Evaluation

Whatever the algorithm used in a recommender system, its final purpose is certain: facilitating users' navigation by recommending to them useful Web resources. The users' trust and satisfaction is the ultimate goal for the design of any recommender system. As a result, ideally, the quality of a recommender system can only be evaluated by its real users. That is, in order to evaluate a recommender system, it must be installed in the real world and real users must be able to use it. We then can ask for the users' feedback in order to evaluate the system. We call this evaluation methodology *user-centred* evaluation. An example of evaluation in this category is given in [3], in which a recommender system for a virtual university information system at the University of Karlsruhe is tested and evaluated by a group of persons, including 6 students, 1 system administrator, 1 secretary, and 5 researchers. They are asked to mark each recommendation presented to them, "right" or "wrong".

Evaluation by the real user is obviously better than evaluation of the algorithm. However, in most cases, this methodology is too ideal to be implemented, for the following reasons – some of which have been demonstrated in the above example:

- As with the design of the rating-based or survey-based recommender system, asking real users to evaluate the system is often considered intrusive. When a recommender system is launched on-line, and users experiencing it are asked to give feedback, most of them do not respond to the request, even though they understand that their feedback will benefit themselves, by helping to improve the quality of the system.

- Organizing a special group of people, for whom it is mandatory to test the recommender system and give feedback to evaluate the system has its own problems. In the first place, it is costly: human resources are expensive and a large number of people are needed, to avoid bias in the test. For example, [3] employs only 13 people for the evaluation, and omits one of the most important portions of users of the system – prospective students. As a result, costs are usually too prohibitive to make extensive testing feasible, especially for a research community. Even for a commercial community, such testing is difficult, as the construction of a recommender system is usually a spiral process. Any recommender system has a great number of parameters to be tested and tuned before launching formally, which cannot be done at one time. In summary, it is unrealistic, both in terms of economy and time, to use this methodology to test a recommender system.

4.1.3 Simulation-based Evaluation

One of the major concerns in designing our recommender system was to avoid user intrusiveness. We would like to make every effort to avoid user involvement, including in the evaluation phase. Thus, evaluation by the real user is not applicable for our system. Rather, we propose a non-intrusive, quantitative evaluation approach.

In the Supervised Machine Learning field, the data set is usually divided into two parts: training data and test data. The former is used to train and build a model, while the latter is used to validate and evaluate the model. Enlightened by that, when we make use of the Web server access log to build our recommender system, we also separate the log data into two parts. The first part, combined with textual content as well as linkage information, is used to build the user profile and the system, as we have described in Section 3, while the second part is kept to test and evaluate the system. This methodology is illustrated in Figure 4.1.

The rationale for this methodology is that what the Web access log records is the user's real navigational behavior. Therefore, entries in the test data log

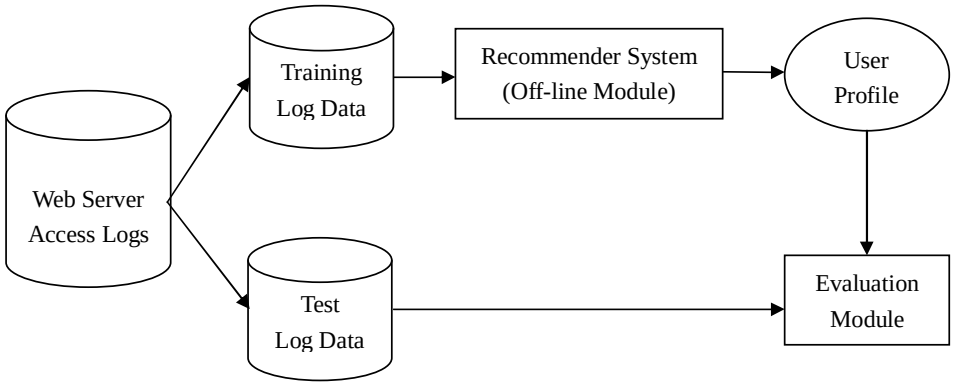

Figure 4.1: Simulation-based Evaluation

can be thought as the behavior of users in the presence of the recommender system. In this scenario, we make an assumption that the user's behavior, in the absence of a Web recommender which has been recorded in the web log, is similar to the user's behavior after we add the recommender on-line. Using this assumption, we can view the test part of the log data as the behavior of some "dummy" users, which simulate the behavior of the real user utilizing the assistance of a recommender system. Hence, we call this the *simulation-based* evaluation approach.

For each Web page in a session in the test log data, our recommender system generates a recommendation list. We can then evaluate the quality of the recommender system by measuring the degree of match between the pages in the recommendation list and in the session (see below for a more detailed description). The evaluation is measured and presented in quantitative format; therefore this approach can be used to compare the effect of different recommender systems, regardless of which Data Mining techniques they adopt: association rule, clustering, or others.

Admittedly, some assumptions have to be made before applying this simulation-based approach. However, it is fast, economical and, most important, automatic and able to avoid any user involvement.

	Negative	Positive
Negative Predicts	a	b
Positive Predicts	c	d

Table 4.1: Confusion Matrix

4.2 Evaluation Metrics

To quantitatively measure the quality of a recommender system, some evaluation metrics have to be employed; however, metrics selection is not a straightforward task. In this section, we first briefly review some metrics for performance measures proposed in the fields of Statistics, Machine Learning, and Information Retrieval, and their application in the recommender system. We then propose our improved metrics.

4.2.1 Metrics for Measuring Recommender System Performance

Some metrics for measuring the performance of information systems, originating from Statistics, Machine Learning, and Information Retrieval have been applied to evaluate recommender systems [58] [50] [101]. Most of them use similar information, which is based on the so called *confusion matrix* [37], as depicted in Table 4.1. The confusion matrix corresponds exactly to the outcomes of a classical statistical experiment. In the matrix, $c + d$ represents those items which should be recommended, while $b+d$ represents what is actually predicted and presented as recommendations by the system. The matrix shows how many of those recommendations actually are correct recommendations (cell d) and how many are not (cell b). The matrix also shows how many of the possible recommendations the system rejected ($a + c$), were correctly rejected (cell a), or should actually have been recommended (cell c).

4.2.2 Performance Metrics from Machine Learning

Performance metrics from Machine Learning are applied mainly to evaluate the performance of the algorithm used in the recommender system – more specifically, to evaluate the ability of the algorithm to learn and build the user

profile and model. The most commonly used performance metrics from Machine Learning are *accuracy* and *coverage*. *Accuracy* is the fraction of correct recommendations out of total possible recommendations (see Formula 4.1). *Coverage* measures the fraction of objects in the search space the system is able to provide recommendations for (see Formula 4.2). We cannot define coverage directly from the confusion matrix, since it only represents information at the level of recommendations.

$$\text{Accuracy} = \frac{\text{Correct Recommendations}}{\text{Total Possible Recommendations}} = \frac{a+b}{a+b+c+d} \tag{4.1}$$

$$\text{Coverage} = \frac{\text{Objects with Recommendations}}{\text{Total Number of Objects}} \tag{4.2}$$

4.2.3 Performance Metrics from Information Retrieval

From the point of view of Information Retrieval, the goal of the recommender system is to help users find items of interest from the set of all available items. To accomplish this task, the most frequently used performance metrics in Information Retrieval are *precision* and *recall*, which are defined in Formulas 4.3 and 4.4, respectively.

$$\text{Precision} = \frac{\text{Correctly Recommended Items}}{\text{Total Recommended Items}} = \frac{d}{b+d} \tag{4.3}$$

$$\text{Recall} = \frac{\text{Correctly Recommended Items}}{\text{Total Useful Recommendations}} = \frac{d}{c+d} \tag{4.4}$$

Often the number of *total useful recommendations* needed for computing recall is unknown, since the whole data collection would have to be inspected. However, instead of the actual *total useful recommendations*, the *total number of known useful recommendations*, that is, $c + d$ in the confusion matrix, is used as an estimate.

Studies have shown that precision and recall are conflicting properties; high precision means low recall, and vice versa [82] [96]. Therefore, an optimal trade-off between precision and recall is required for an Information Retrieval system.

4.2.4 Recommendation Accuracy and Shortcut Gain

As mentioned before, a Web recommender system can be utilized to recommend either products, such as books, movies, and music CDs; or on-line resources, such as Web pages or on-line activities. Our system can be seen as an improved Web usage recommender system mainly designed to be used for the latter purpose. In this scenario, we recognize that the quality of a recommender system depends on whether the recommendation can accurately predict users' information needs, and assist them to fulfill their needs as quickly as possible. Consequently, we propose two measurement metrics: *Recommendation Accuracy* and *Shortcut Gain*. *Recommendation Accuracy* measures the degree to which the recommendation given by a recommender system accurately predicts the users' needs and interest. In fact, this metric is identical with the Precision given by Formula 4.3. If the recommendation matches with users' needs and interests, the user will probably like it. Thus, Recommendation Accuracy can be used to measure the possibility that the user will like the recommendation. This measure indicates the trust of the user in the system. It is impossible for a system with a lower Recommendation Accuracy to gain the trust of the user. Further, if the system can recommend a resource the user is interested in, but would have to make considerable effort to reach, it is highly probable the user's trust in the system would be strengthened. Therefore, a new metric, *Shortcut Gain*, is introduced in this thesis, and is used to measure the ability of a recommender system to reduce the navigation effort needed by users to fulfill their information needs. More precisely, we measure the *Recommendation Accuracy* and the *Shortcut Gain* as described below.

Recommendation Accuracy is the ratio of correct recommendations among all recommendations, and the correct recommendation is the one that appears in the suffix of a session from which the prefix triggers the recommendation. As an example, consider that we have S visit sessions in the test log. For each visit session s, we take each page p and generate a recommendation list *R(p)*. *R(p)* is then compared with the remaining portion of s (i.e., the suffix of s). We denote this portion *T(p)* (T stands for Tail). The recommendation

accuracy for a given session would be how often *T(p)* and *R(p)* intersect. The general formula for *recommendation accuracy* is defined as:

$$RecommendationAccuracy = \frac{\sum_s \frac{\left|\bigcup_p (T(p) \bigcap R(p))\right|}{\left|\bigcup_p R(p)\right|}}{S}$$

The *Shortcut Gain* measures how many clicks the recommendation allows users to save if the recommendation is followed. Suppose we have a session a, b, c, d, e, and at page b, the system recommends page e; then, if we follow this advice, we would save two hops (i.e., pages c and d). An issue arises in measuring this Shortcut Gain when the recommendation list contains more than one page in the suffix of the session. Should we consider the shortest gain or the longest gain? To solve this problem, we opted to distinguish between *key* pages and *auxiliary* pages. A *key* page is a page that may contain relevant information, and in which a user may spend some time. An *auxiliary* page is an intermediary page used for linkage, and in which a user spends a relatively short time. In our experiment, we use a threshold of 30 seconds to make this distinction. Given these two types of pages, a Shortcut Gain is measured as being the smallest jump gain towards a *key* page that has been recommended. If no *key* page is recommended, then it is the longest jump towards an *auxiliary* page. The set of pages in the session we go through with the assistance of the recommender system is called the improved session *s'*. For the total S visit sessions in the test log, *Shortcut Gain* can be computed as:

$$ShortcutGain = \frac{\sum_s \frac{|s|-|s'|}{|s|}}{S}$$

In addition, we would like to compute the *Coverage* of a recommender system, to test the consistency of system performance. *Coverage* measures the ability of a system to produce all pages that are likely to be visited by users. The concept is similar to the Recall in Information Retrieval. *Coverage* is defined in our system as:

$$RecommendationCoverage = \frac{\sum_s \frac{\left|\bigcup_p (T(p) \bigcap R(p))\right|}{\left|\bigcup_p T(p)\right|}}{S}$$

4.3 Experimental Results

Using the definitions of Recommendation Accuracy, Shortcut Gain, and Coverage, we evaluated our recommendation framework on the University of Alberta Department of Computing Science Web site. In this section, we report and discuss the evaluation results.

In our experiment, we collected Web server access logs for 8 months (Sept. 2002 – Apr. 2003), and partitioned the data into months. We also collected and stored content data, as well as structure data of the Web site. On average, each monthly partition contains more than 40,000 pages, resulting in, on average, 150,000 links between them. The log for each month averaged more than 200,000 visit sessions, which generated an average of 800,000 missions per month. The modified DC-tree content clustering algorithm generated about 1500 content clusters, which we used to identify the missions per month.

Given the data partitioned per month as described above, we adopt the following empirical evaluation: one or more months' data is used for building our models (i.e., training the recommender system), and the following month or months for evaluation. The reason we divide the data based on a time frame (months), rather than use standard cross-validation in Machine Learning on the data set, is that we want to measure the prediction ability of our system for the future, rather than merely the past. Moreover, the Web site evolves over time. More specifically, given a session s from a month m, if the recommender system, based on data from month $m-1$ and some prefix of the session s, can recommend pages p_i that contain some of the pages in the suffix of s, then the recommendation is considered accurate. Moreover, the distance in the number of clicks between the suffix of s and the recommended page p_i is considered a gain (i.e., a shortcut).

We evaluate the Recommendation Accuracy, Shortcut Gain, and Coverage of our recommender system, which makes use of usage, content, and structure information (referred to as *Hybrid123*). For the purpose of comparison, we also implement a pure Web Usage recommender system (referred to as *Usage*), based on [17] using the association rule technique; and a pure Content-

based recommender system (referred to as *Content*), based on [67] using the clustering technique. The *Usage* system works as follows: an efficient association rule algorithm [17] is applied to the access logs to generate a set of rules. Whenever the pages in the antecedent of a rule have appeared in the user's current session, those pages in its consequence are recommended. For the *Content* system, all pages recorded in the access logs (referring to all pages having once been accessed) are grouped into clusters solely based on their textual content, using a content clustering algorithm [67]. If one or more pages in a cluster have been visited in the user's current session, the pages in the same cluster are selected to be recommended. To avoid supplying a very large list of recommendations, and to make the comparison in the same context, we apply the same limitation of recommendations number of our system on the two. That is, we limit the number of recommendations in the two systems $\sqrt{n}$ best recommendations (n is the number of links), with a maximum of 10. To pick up the best recommendations, for all rules with the same antecedent, recommendation candidates in *Usage* are ranked by the Confidence scores of the individual rules; while in *Content*, candidates are ranked by the cosine similarity with the current page. The Recommendation Accuracy and Shortcut Gain of the three systems are depicted in Figures 4.2 and 4.3, respectively. In the experiment, we vary the Recall to test the trend and consistency of the system quality. We have to point out that it is very difficult to locate parameters to cause the three systems to perform with the exact same Recall values. Therefore, what we do in the experiment is try different parameters for individual systems, and report the best results in a series of similar Recall values.

As expected, the Accuracy decreases when the Recall is increased from all three systems. However, *Hybrid123* is consistently the best among the three systems, superior to *Usage* by at least 30% – while *Usage* always ranks second. The last-place ranking of *Content* justifies our argument that using content information only is not sufficient to build a Web recommender system. This is mainly because having pages with similar content does not guarantee that they are related to a goal. We can conclude from the figure that usage

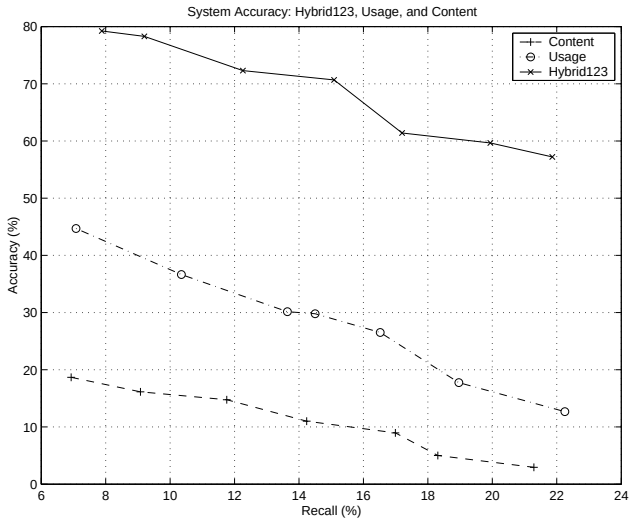

Figure 4.2: System Performance Comparison: Recommendation Accuracy

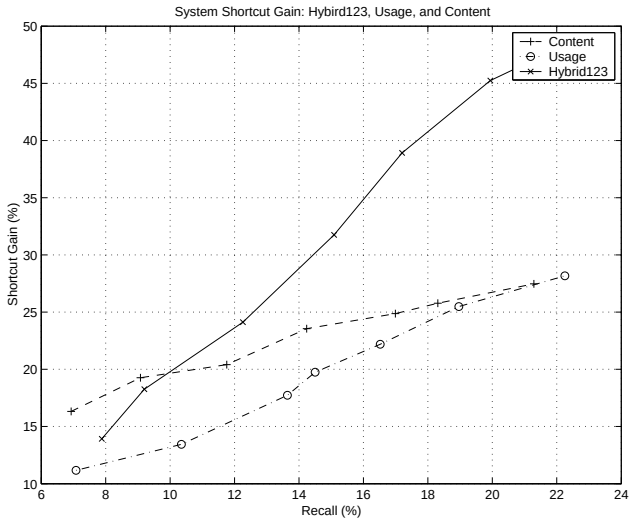

Figure 4.3: System Performance Comparison: Shortcut Gain

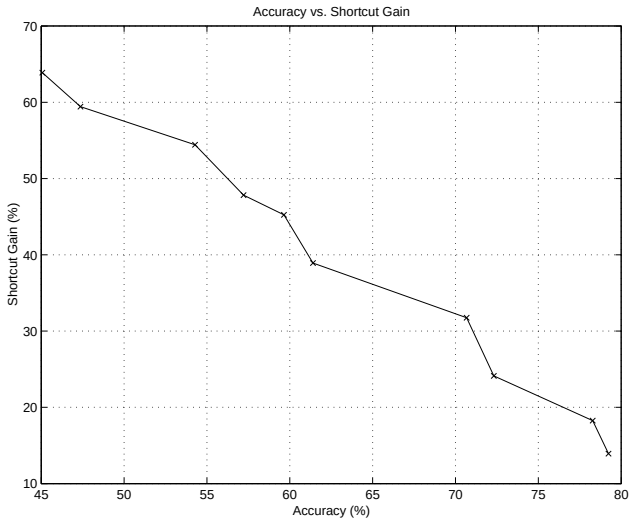

Figure 4.4: Shortcut Gain vs. Recommendation Accuracy

information is a much better alternative to content in contributing to recommendation accuracy, while attaching content to usage could further improve system accuracy.

From Figure 4.3, we can see that in the low boundary, the Shortcut Gain of *Content* is the best of the three systems, and the other two are close. With the increase of Recall, the Shortcut Gain of all three systems continues to improve, but in different degrees. *Hybrid123* can achieve an increasingly superior Shortcut Gain to that of *Usage*, and exceeds *Content* after Recall is larger than about 10%. The major reason that the Shortcut Gain improvement of *Content* is lowest is that with the increase of Recall, more and more pages containing only the same keywords, but without any logical relationship are selected to be recommended. This further proves that only recommending pages with similar content does not necessarily assist the user to fulfill their information needs.

Combining Figures 4.2 and 4.3, we can also see the relationship between Recommendation Accuracy and Shortcut Gain in our system (*Hybrid123*), as depicted in Figure 4.4. It shows that Recommendation Accuracy is inversely proportional to the Shortcut Gain. Our study draws the same conclusion from the *Usage* and *Content* system. We argue that this is an important property of a Web recommender system. Therefore, how to adjust and balance

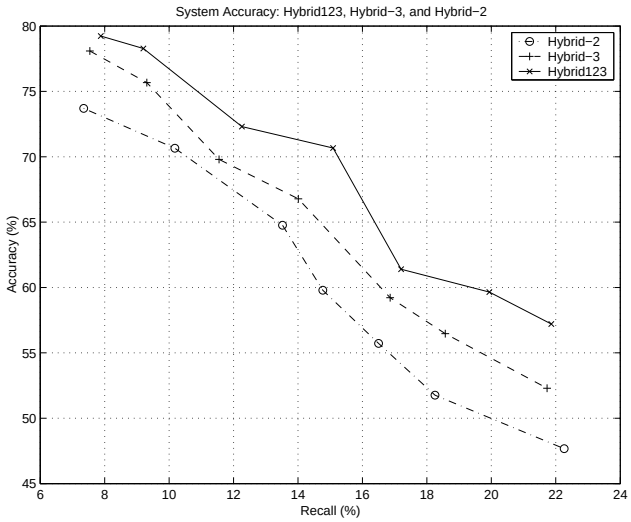

Figure 4.5: *Hybrid123*, *Hybrid-3*, and *Hybrid-2*: Recommendation Accuracy

between Recommendation Accuracy and Shortcut Gain, in order for a Web recommender system to achieve the maximum benefit, is a question that should be investigated. Some web sites, e.g., those with high link density, may favor a recommender system with high Recommendation Accuracy, while others may favor a system with high Shortcut Gain.

In our next experiment, we illustrate the advantage of incorporating Web content and Web structure information in our system. To do so, we implemented an additional two recommender prototypes. The first is similar to *Hybrid123* but is stripped from its connectivity information channel. That is, we do not make use of linkage information to augment and improve the user profile built on usage and content information. We name this hybrid recommender *Hybrid-3*. The second is also a similar system to *Hybrid123*, but does not make use of content information to identify a mission. Rather, the user profile in the system is built upon traditional transactions identified according to the approach in [17]. Then, the user profile is improved with structure information, as with *Hybrid123*. This hybrid system is called *Hybrid-2*. The Recommendation Accuracy and Shortcut Gain of the three systems are depicted in Figures 4.5 and 4.6.

Figure 4.5 shows the Recommendation Accuracy of the three contenders. The consistent best performance of *Hybrid123* illustrates the validity of us-

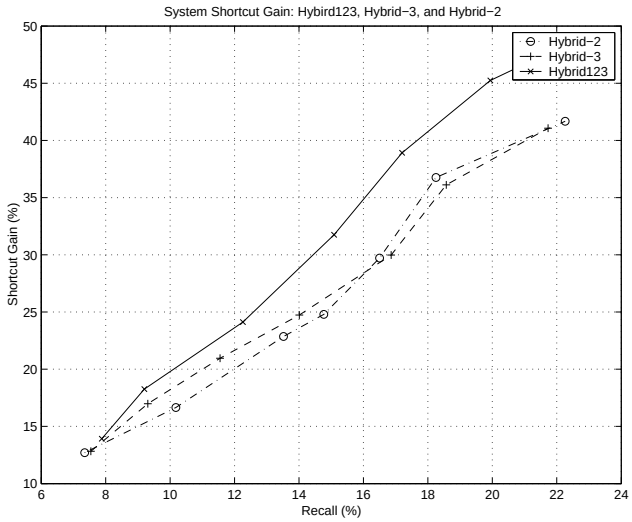

Figure 4.6: *Hybrid123*, *Hybrid-3*, and *Hybrid-2*: Shortcut Gain

ing content and connectivity information to improve recommendations in our recommender system, and also indicates that content is more useful for recommendation accuracy improvement. The Shortcut Gains of the three systems are depicted in Figure 4.6. We notice that with the increase of Coverage, *Hybrid123* can achieve an increasingly superior Shortcut Gain compared to both *Hybrid-3* and *Hybrid-2*, while those two systems maintain a similar performance in terms of Shortcut Gain. This further verifies our justification for using distinct information channels in building a hybrid recommender system, and shows that content and structure information make a similar contribution to the improvement in Shortcut Gain. This experiment also proves that our mission-based model is better than traditional transaction-based model.

Chapter 5

The Case of VIVIDESK

In addition to the University of Alberta Department of Computing Science Web site data, we also apply and test our system with another data set provided by the VIVIDESK system. VIVIDESK[1] is a commercial system developed by the Centre of Health Evidence at the University of Alberta, as a gate to a multitude of applications and on-line resources, and is used by hospital personnel. The VIVIDESK system tracks and records in its log the on-line activity and requests of those health care providers that are using the system. The VIVIDESK data was originally considered in our intention to further test and verify the effectiveness and applicability of our recommender system. However, there is no favorable content and structure information available in the system. On the other hand, it has a specific session-based activity log which records details about user accesses to on-line resources via different applications pre-included in the system. Based on this idiosyncratic data set, we extend our previous work, in particular to identify missions to build the user profile. We discuss this in detail in the following sections.

5.1 VIVIDESK Data

First, we will give a brief introduction concerning this data set, and discuss the difference between it and the generic Web server access logs.

The VIVIDESK system is a new and powerful tool provided by the Centre of Health Evidence (CHE), University of Alberta Faculty of Medicine and

[1] http://www.vividesk.com/

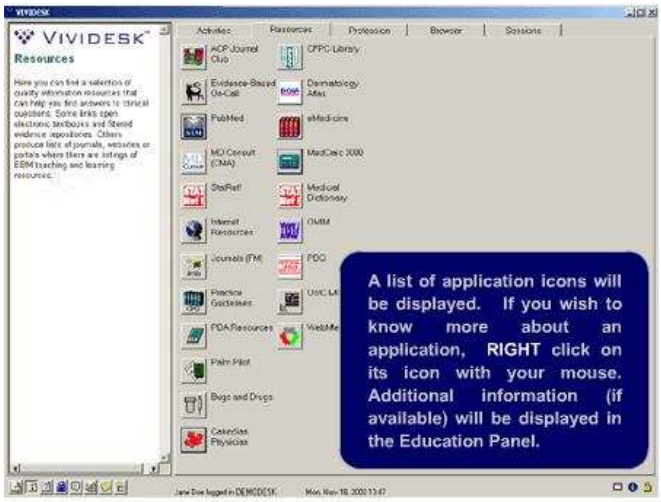

Figure 5.1: A Snapshot of the VIVIDESK Desktop

Dentistry, to assist health care providers to fulfill their knowledge acquisition needs. In a hospital, many unusual medical situations occur, some of which will be beyond the practitioner's knowledge. Ultimately, a decision about the best care for the patient must be made. Faced with such a situation, health care providers traditionally would rely upon their own acquired knowledge and experience, try to find the information in a pocket reference book, or consult with a colleague or superior to obtain the information needed to make a sound decision. These methods are quite common and will continue to be used in health care settings. The CHE electronic desktop, VIVIDESK, however provides another efficient and effective alternative. A VIVIDESK desktop is a set of pre-selected applications which are believed to be useful for helping physicians and nurses to make their decisions in the clinical environment. The applications are selected and assembled into the VIVIDESK desktop by an editorial committee, composed of experienced physicians, librarians, and nurses. The editorial committee evaluates existing and potential resources according to their usefulness to health professionals. The application could be bibliographic databases such as *MEDLINE*, Microsoft Office package, or a hub of a collection of Web resources for a special hospital task. Figure 5.1 shows a snapshot of a VIVIDESK desktop.

When health care providers use the desktop, it tracks their on-line activities

and requests, and records how the individual applications are used and which web pages are visited in the so called VIVIDESK logs.

The VIVIDESK log is similar to the Web server access log in terms of some basic information recorded in both systems, such as client address and request date and time, but it has its own distinct properties:

- VIVIDESK records both application usage and Web access usage, and it records the latter within the former. VIVIDESK records how individual applications which have been pre-organized are used. When utilizing such an application, a user can access the Internet, either by following the links, or launching a browser, both provided within the application. This Web access is also recorded. Moreover, the relationship between applications and visited Web resources – that is, which resources are visited from which particular application – is also recorded.

- A Web server access log records only users' requests to a particular Web server, but VIVIDESK records all Web requests from users, not limited to any specific Web site.

- In the VIVIDESK system, any users are required to login and logout. Thus, the VIVIDESK log is a session-based access log, having entries identified by users. As a result, there is no need to identify user and session for this access log. Please note that although users login and logout information is recorded in the log, user privacy is still protected when this dataset is used to build our recommender system, as our system never refers to a specific user when giving recommendations.

5.2 Mission Identification on VIVIDESK Data

The mission identification approach discussed in Section 3.2.2 relies on the availability of the textual content of web pages, which could not always be provided. For instance, no content information about the pages visited in the VIVIDESK system are available to us. In fact, the Web pages recorded in the VIVIDESK system come from different Web sites, and a large number of

them are dynamically generated, which makes retrieving page textual content difficult or even impossible. Can we identify missions in the absence of textual content? In this section, we answer the question by investigating alternative approaches to identifying missions from the VIVIDESK access log.

We recognize that in the VIVIDESK system, all Web pages are requested in individual applications, while these applications are carefully selected and assembled in the system. A common usage of the VIVIDESK desktop is as follows: a medical professional logs into the system to look for some resources to aid in making a decision, and logs out when the necessary information has been found. During a log-in session, the physician clicks to open one or more applications on the desktop. Within each application, the physician could follow the link(s) to access a Web resource, or start an on-line search – both provided in the application – in case the expected information is not available in the application. More often than not, physicians open more than one application at the same time. The following example illustrates a typical log of a session in the VIVIDESK system:

Application 1: a-b-a-d-e

Application 2: f-g

Application 2: h-a-i

Please note that the log in the VIVIDESK system is organized in *Application-Web Resource* format. That is, the system tracks and records the sequence of applications in a log-in session, and the sequences of Web pages invoked within individual applications. In a typical Web server log, the same session would be recorded as: a-b-f-a-g-d-h-a-i-e.

Although there is no textual content available for us to identify the content coherent missions as we desired, we work out another approach to identify missions which would portray users' concurrent information needs in the VIVIDESK log-in session. We argue that all Web pages invoked within an application could be highly related by virtue of their functionality, because each application is carefully scrutinized and selected by experts to fulfill a specific information need of health care providers. We therefore identify and organize all Web pages invoked within an application as a mission. Thus, for

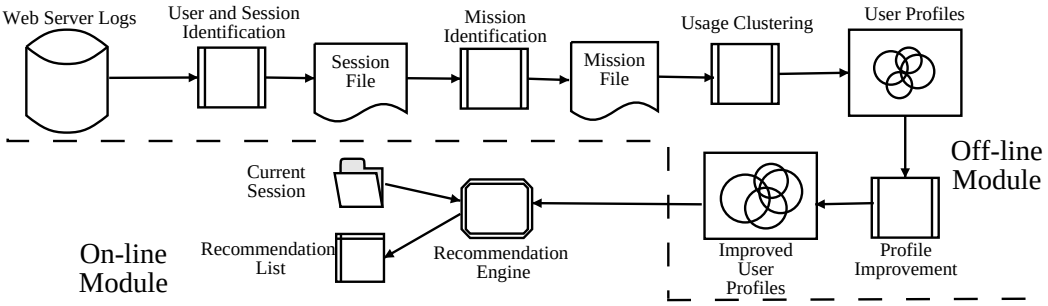

Figure 5.2: Generalized System Architecture

the example above, our system identifies three missions as follows:

Mission 1: a-b-a-d-e

Mission 2: f-g

Mission 3: h-a-i

Our preliminary experiments (see Section 5.3) show that this is a good approach to identifying missions for VIVIDESK data; moreover, it generalizes our notion of *mission*. In fact, the notion of *mission* is proposed to identify concurrent information needs during visit sessions. Generally, the mission can be discovered based on the content similarity among pages visited during that session. However, this concept is suitable and applicable in a context without such information, to sites containing pages that are not content-rich, or pages that are highly related by the virtue of their functionality rather than content (e.g., a sequence of steps that is part of an on-line dynamic application). Here, our work on the VIVIDESK data highlights the importance of having application-related logs for mission identification. Admittedly, how to identify missions in different environments should be further discussed and expanded. With this understanding of mission, the architecture of our recommender system can be generalized as depicted in Figure 5.2. User and Session Identification as well as Profile Improvement in the architecture, are optional, depending on the data available. For instance, since there is no structure information available in the VIVIDESK data, we do not include the user profile improvement sub-system when we apply our system to the VIVIDESK data.

In addition, we found that the VIVIDESK system records in its logs keystrokes made by users, along with access information, within individually

invoked applications. For example, the system records the text entered by the user in HTML form input fields. These text data, while not the real content of Web resources, can also be associated with the visited pages and used to separate sessions into missions. Specifically, we attempt to merge our former identified missions (referred to as *App-Mission*), solely based on application, if they are "content" similar. The "content" similarity is judged by the text body input in the individual applications. Using this approach, we can identify another type of mission, which we name *Text-Missions*. With the two different missions, we can build two variations of our system for the VIVIDESK desktop, referred to as *App-Mission* and *Text-Mission*, respectively.

5.3 Experiments on VIVIDESK Data

In this session, we test and evaluate the performance of our system on the VIVIDESK data, with the emphasis on verifying the advantage of our mission-based model in the absence of content information. In the experiment, we use the same metrics – that is, Recommendation Accuracy and Shortcut Gain – to test the performance of both *App-Mission* and *Text-Mission*; and we also vary the Coverage to reveal the performance tendency. For the purpose of comparison, we also implemented another variation of our system which uses traditional transactions, as defined in [21], to build user profiles (referred to as *Tran*).

The experimental results are depicted in Figures 5.3 and 5.4. We see that *App-Mission* is able to achieve a higher Recommendation Accuracy than is simple transaction identification, but leads to a lower Shortcut Gain. However, because we can achieve a much higher Recommendation Accuracy with a slight loss of Shortcut Gain, we can be confident that mission identification is a better model for user navigational behavior. The reason *App-Mission* leads to a lower Shortcut Gain is that we identify missions based solely on invoked applications, with the absence of content. However, users may need more than one application to fulfill one information need. Thus, identifying missions based on applications may break some interrelationship between web resources

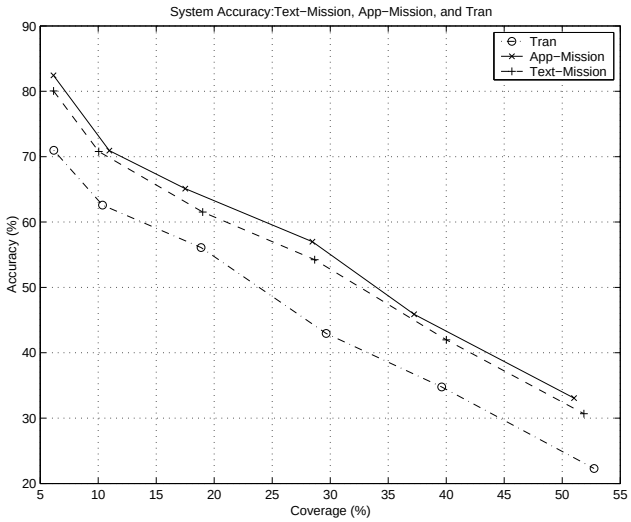

Figure 5.3: System Performance on VIVIDESK Data: Recommendation Accuracy

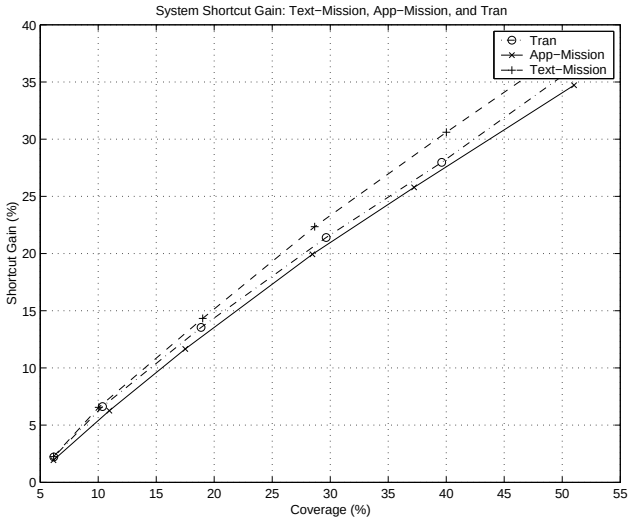

Figure 5.4: System Performance on VIVIDESK Data: Shortcut Gain

across applications. However, this can be offset when we use the text entered by the user to merge the mission identification, as we do in *Text-Mission*. In fact, the Shortcut Gain achieved by Text-Mission is even higher than that achieved by the transaction-based approach. However, the Recommendation Accuracy of *Text-Mission* is slightly lower than *App-Mission*.

5.4 Novice vs. Pragmatic Users in the VIVIDESK System

All our experiments presented so far (for both the VIVIDESK data and the UofA CS Department Web site data) are based on month-by-month partitions; that is, the data collection is partitioned by month. One or more months' data is used for training the system, and the following month or months for evaluation. We notice that the VIVIDESK log is a session-based access log, which also records the log-in number of each login-in session for individual users. Thus, we know how many times a user had logged in and used the system before the current login. This gives us some hints about what type of user s/he is – a pragmatic user, or a novice one. A doctor who has logged into the VIVIDESK desktop more than 100 times would be much more familiar with the system than a new user. As a result, the former would likely be better informed as to where the resources are to fulfill a specific information need, and be able to reach them faster, with fewer mistakes than the latter. Arguably, selecting only the portion of the VIVIDESK log belonging to the pragmatic user, to train our recommender system could further improve recommendation quality, in particular for the novice user. To verify this point, we perform another experiment. In this experiment, rather than dividing the dataset by time, we divide it into two parts, according to the login numbers. More specifically, we rank all the log-in sessions in the VIVIDESK log decreasingly, according to their login numbers. We then cut this ranked dataset in the middle. We consider that log-in sessions in the first group are performed by pragmatic users, while log-in sessions in the second group are records of novice users. We use the first part of the data to train our system, and the second part

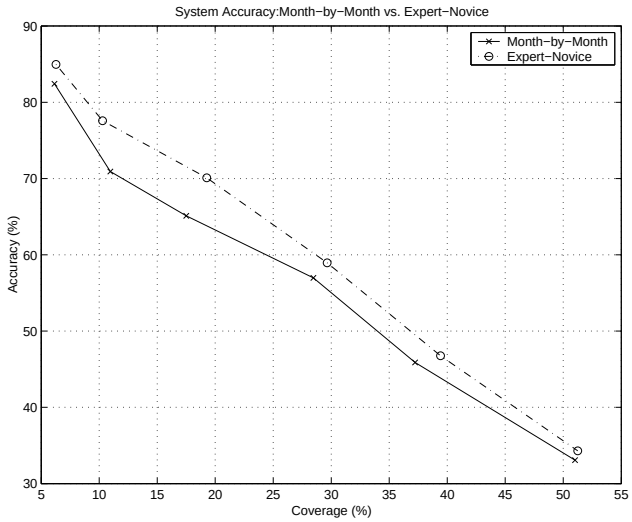

Figure 5.5: Accuracy: Month-by-Month vs. Expert-Novice

for testing (referred to as *Expert-Novice*). We assume here that novice users will eventually reach the resources they are looking for, and our recommender system would then provide them with shortcuts. Figures 5.5 and 5.6 show the results, when the same approach used in *App-Mission* is adopted for mission identification. For the purpose of comparison, we re-depict the experimental results of *App-Mission* which are originally presented in Figures 5.3 and 5.4 (here *App-Mission* is referred to as *Month-by-Month*).

Both figures justify our argument to exclusively use the access history of the pragmatic user to achieve better recommendation. Indeed, both Recommendation Accuracy and Shortcut Gain show improvement.

This experiment could also be done using the UofA CS Web site data, by considering that users of month m are more Pragmatic than users of month $m-1$, if these users are the same.

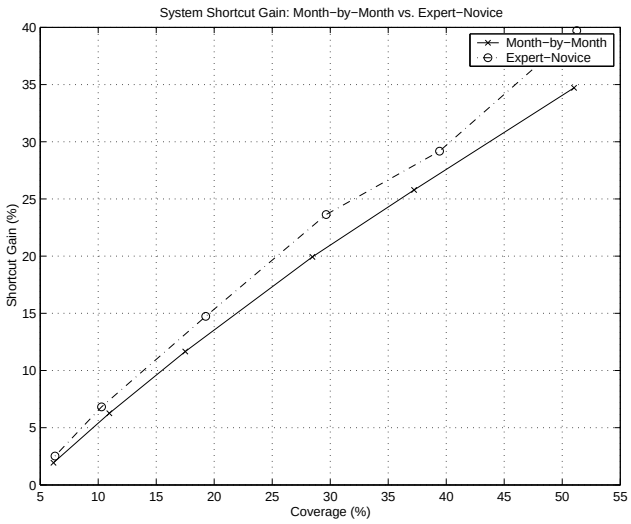

Figure 5.6: Shortcut Gain: Month-by-Month vs. Expert-Novice

Chapter 6

Conclusion and Future Work

In this thesis, we present a framework for a hybrid web recommender system, which utilizes, when possible, the content and connectivity of Web pages, in addition to usage history. These different information channels are used and combined in several ways and stages. First, we propose a novel mission-based navigation model. The notion of mission is proposed in order to identify Web users' concurrent information needs during their on-line navigation, and we make no assumption concerning the sequence in which these needs are fulfilled. Thus, this mission-based model is better and more practical than the traditional, widely used transaction-based model in describing the navigational behavior of the real on-line user. In our framework, we use the textual content of Web pages to identify missions from the Web access history; we then use the PageGather algorithm to cluster the mission file. Each of the resulting clusters possesses two characteristics: it is usage cohesive and content coherent. These clusters are, therefore, suitable to be used as the user's navigational profile, to aid in fulfilling their information needs regarding a specific topic. We then combine the linkage information and content information to augment and then prune the mission clusters, to include those pages neighboring nodes in the individual clusters and still having the same focused topic. Thus, our system provides an opportunity for these rarely visited or newly added pages to be recommended. The Web structure information is also used to compute the Hub and Authority scores of pages, according to the HITS algorithm, in individual clusters. These scores are used to rank the recommendation candi-

date, with the more important items being recommended. Our experiments show that the combination of usage, content, and structure of data in a Web recommender system has the potential to improve the quality of the system, as well as to keep the recommendations up-to-date. Indeed, our framework has provided an initial attempt and demonstration of combining these channels for recommendation improvement. In addition, we emphasize using and combining these distinct information channels off-line, which provides our framework with another benefit: recommendation efficiency, i.e., reducing low latency for recommendation.

However, there are various ways to use and combine these different information channels to improve system quality, either in the on-line module or the off-line module of a Web recommender system. We discuss several cases which take advantage of structure information in a recommender system, as examples. We have concluded that, according to our experiments, Recommendation Accuracy is inversely proportional to the Shortcut Gain for a Web recommender system. Therefore, these two metrics should be adjusted and balanced for any specific Web recommender system, in order to achieve the maximum benefit. The Web site structure as well as linkage information, would be helpful in accomplishing that goal. Generally speaking, for a Web site hierarchy such as that of the UofA CS Web site, the upper layers are more deliberately designed than are the lower layers. In addition, users are generally familiar with how to reach the pages they are looking for in these upper layers and, as a result, they would prefer a recommendation with high Shortcut Gain. On the other hand, in the lower layers, where Web resources may not be well-organized and there are fewer user visits, users would be more likely to prefer a recommendation with high Recommendation Accuracy. In addition, when a user stays on a page with high link density and needs advice, s/he usually favors a recommendation with high Recommendation Accuracy; otherwise, a recommendation leading to a big jump is preferred. Thus, the structure information could be used to adjust and balance Recommendation Accuracy and Shortcut Gain dynamically for the purpose of maximizing user trust. Furthermore, the Web site structure information could be used to pro-

vide better parameter adjustment in the system. For example, existing usage-based Web recommender systems using the association rule algorithm make use of the same Support and Confidence score on all visit sessions to generate rules to be used for recommendation. As we have discussed, different layers of a Web site have diverse visit frequency and, generally speaking, hierarchies tend to get more hits near the root, while pages in lower layers are more likely to concentrate on a specific topic. Thus, we would like to provide different rules in which items from different layers have different Support and Confidence thresholds. The simplest situation might be to increase Support but decrease Confidence to generate rules for sessions in which items belong to the upper layers of the site hierarchy; but lower Support and raise Confidence for the sessions involving items in the lower layers. From this discussion, we can conclude that there are many possible and potential ways available to utilize the distinct information channels in a recommender system; unfortunately, the best one is not yet evident. It is possible to combine the channels in a different but more effective way than is done in our system, and our future work in this area will include investigating and comparing the different methods.

We would also like to utilize the user-centred method to further validate our previous simulation-based evaluation. Data Mining technology was initially proposed to help people deal with the sea of data available; therefore, a fully automatic system was always more preferred. Until recently, it has been accepted that it is virtually impossible for a fully automatic system to fulfill the necessary tasks based on the current development of science and technology. To provide for the most effective data mining, the user must be allowed to be front and centre in the mining process. A proper division of labour between computers and humans is a new topic and challenge in the Data Mining field. In our Web recommender system evaluation, we have recognized (Section 4.1.2) that user-centred evaluation is a superior way to evaluate a Web recommender system. To make full use of the advantages of automatic data mining, we would combine the user-centred and simulation-based evaluations in building and evaluating a recommender system. The simulation-based evaluation is used to select the best parameters to be used and to pre-test the

performance of the system. When we are confident with the lab results of the system, real users will be invited to further test and evaluate it.

Bibliography

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *The ACM SIGMOD International Conference On Management of Data*, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *The 20th VLDB Conference*, 1994.

[3] Maximillian Jahn Andreas Geyer-Schulz, Michael Hahsler. Educational and scientific recommender systems: Designing the information channels of the virtual university. *International Journal of Engineering Education*, 17(2), 2001.

[4] M. Balabanovic. Fab: Content-based, collaborative recommendation, 1997. Communications of the ACM, 40(3).

[5] Krishna Bharat and Monika R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 104–111, Melbourne, AU, 1998.

[6] D. Billsus and M. Pazzani. A hybrid user model for news story classification. In *The Seventh International Conference on User Modeling*, 1999.

[7] D. Billsus and M. Pazzani. User modeling for adaptive news access, 2000. User-Modeling and User-Adapted Interaction 10(2-3), 714-720.

[8] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *The 14th Annual Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[9] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *The 7th World-Wide Web Conference*, 1998.

[10] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. In *The 9th International World-Wide Web Conference (WWW-9)*, 2000.

[11] R. Burke. Hybrid recommender systems: Survey and experiments, 2002. User Modeling and User-Adapted Interaction. 12(4), pages 331-370.

[12] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of navigation patterns on a web site using model based clustering, 2000. Technical Report MSR-TR-00-18, Microsoft Research.

[13] L. Catledge and J. Pitkow. Characterizing browsing behaviors on the world wide web, 1995. Computer Networks and ISDN Systems, 27(6).

[14] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *Proceedings of the 7th International World Wide Web Conference*, 1998.

[15] Arbee L.P. Chen, Yi-Hung Wu, and Yong-Chuan Chen. Enabling personalized recommendation on the web based on user interests and behaviors. In *the 11th International Workshop on Research Issues in Data Engineering*, 2001.

[16] Jiyang Chen, Lisheng Sun, Osmar R. Zaïane, and Randy Goebel. Visualizing and discovering web navigational patterns. In *The Seventh ACM SIGMOD International Workshop on the Web and Databases (WebDB 2004)*, 2004.

[17] M.-S. Chen, J.-S. Park, and P.S. Yu. Efficient data mining for path traversal patterns. *IEEE Transactions on Knowledge and Data Engineering*, 10(2):209–221, 1998.

[18] D. Cheung, J. Han, V. T. Ng, A. W. Fu, and Y. F. A fast distributed algorithm for mining association rules. In *The 1996 International Conference on Parallel and Distributed Information Systems (PDIS '96)*, 1996.

[19] E. H. Chi. Improving web usability through visualization. *IEEE Internet Computing*, 6(2), 2002.

[20] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *1999 Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

[21] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

[22] J. Dean and M. R. Henzinger. Finding related pages in the world wide web. In *The 8th International World-Wide Web Conference (WWW-8)*, 1999.

[23] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. Advances in knowledge discovery and data mining, 1996. AAAI/MIT Press.

[24] P. W. Foltz. Using latent semantic indexing for information filtering. In *The 1990 Conference on Office Information Systems*, pages 40–47, 1990.

[25] Xiaobin Fu, Jay Budzik, and Kristian J. Hammond. Mining navigation history for recommendation. In *Intelligent User Interfaces*, pages 106–112, 2000.

[26] Johannes Fürnkranz. Web structure mining - web structure mining exploiting the graph structure of world-wide web. *ÖGAI Journal*, 21(2), 2002.

[27] D. Goldberg, D. Nichols, and B. M. Oki. Using collaborative filtering to weave an information tapestry, 1992. Communications of the ACM, 35(12), 61-70.

[28] R. H. Guttman. Merchant differentiation through integrative negotiation in agent-mediated electronic commerce, 1998. Master's Thesis, School of Architecture and Planning, Massachusetts Institute of Technology.

[29] John Riedl J. Ben Schafer, Joseph Konstan. Recommender systems in e-commerce. In *The First ACM Conference on Electronic Commerce*, pages 158–166, 1999.

[30] Authony Jameson, Joseph Konstan, and John Riedl. Conference tutorial notes: Ai techniques for personalized recommendation. In *Conference Tutorial Notes: Eighteenth National Conference on Artificial Intelligence*, 2002.

[31] T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web. In *The 15th International Conference On Artificial Intelligence*, 1997.

[32] Thorsten Joachims, Dayne Freitag, and Tom M. Mitchell. Web watcher: A tour guide for the world wide web. In *IJCAI (1)*, pages 770–777, 1997.

[33] B. Kahle. Archiving the internet. Scientific American, March 1997.

[34] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in vlsi domain. In *ACM/IEEE Design Automation Conference*, 1997.

[35] L. Kaufman and P. J. Rousseeuw. Finding groups in data: an introduction to cluster analysis, 1990. John Wiley & Sons.

[36] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[37] R. Kohavi and F. Provost. Special issue on applications of machine learning and the knowledge discovery process, 1998. Machine Learning 30(2/3) 1998.

[38] Ron Kohavi, Brij M. Masand, Myra Spiliopoulou, and Jaideep Srivastava (editors). Workshop on mining web log data across all customers touch points (webkdd '01), 2002.

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

[40] Raymond Kosala and Hendrik Blockeel. Web mining research: A survey. In *The ACM SIGKDD Explorations*, 2000.

[41] B. Krulwich. Lifestyle finder: Intelligent user profiling using large-scale demographic data, 1997. Artificial Intelligence Magazine, 18(2), 37-45.

[42] Wai Lam and Chao Yang Ho. Using a generalized instance set for automatic text categorization. In *The 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998.

[43] K. Lang. Newsweeder: Learning to filter news. In *The 12th International Conference on Machine Learning*, pages 331–339, 1995.

[44] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *The ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

[45] David D. Lewis, Robert E. Schapire, Jame P. Callan, and Ron Papka. Training algorithms for liner text classifiers. In *The 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996.

[46] Henry Lieberman. Letizia: An agent that assists web browsing. In *1995 International Joint Conference on Artificial Intelligence*, 1995.

[47] Henry Lieberman. Autonomous interface agents. In *Proceedings of the ACM Conference on Computers and Human Interface, CHI-97*, Atlanta, Georgia, 1997.

[48] Henry Lieberman. Why surf alone? exploring the web with reconnaissance agents. In *WebNet Conference, Association for the Advancement of Computing in Education*, 1998.

[49] C. Lin, S. Alvarez, and C. Ruiz. Collaborative recommendation via adaptive association rule mining, 2000. Workshop on Web Mining for E-Commerce – Challenges and Opportunities (WebKDD '00).

[50] Weiyang Lin, Sergio A. Alvarez, and Carolina Ruiz. Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery*, 6(1), 2002.

[51] Bing Liu, Wynne Hsu, and Yiming Ma. Pruning and summarizing the discovered associations. In *The 6th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-99)*, 1999.

[52] B. Masand and M. Spiliopoulou (editors). Workshop on web usage analysis and user profiling (webkdd '99), 1999.

[53] O. A. McBryan. Genvl and wwww: Tools for taming the web. In *The 1st World-Wide Web Conference*, 1994.

[54] Alberto Mendelzon, George Mihaila, and Tova Milo. Querying the world wide web. In *The 1996 Conference on Parallel and Distributed Information Systems*, 1996.

[55] B. Miller, J. Riedl, and J. Konstan. Experiences with grouplens: Making usenet useful again. In *The 1997 Usenix Winter Technical Conference*, 1997.

[56] Dunja Mladeni and Marko Grobelnik. Feature selection on hierarchy of web documents. *Decision Support Systems*, 35(1), 2003.

[57] Dunja Mladenic. Feature subset selection in text-learning. In *European Conference on Machine Learning*, 1998.

[58] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Discovery and evaluation of aggregate usage profiles for web personalization. *Data Mining and Knowledge Discovery*, 6(1), 2002.

[59] Bamshad Mobasher, Honghua Dai, Tao Luo, Yuqing Sun, and Jiang Zhu. Integrating web usage and content mining for more effective personalization. In *EC-Web*, pages 165–176, 2000.

[60] Bamshad Mobasher, Eui-Hong Han, George Karypis, and Vipin Kumar. Hypergraph based clustering in high-dimensional data sets: A summary of results. *IEEE Bulletin of the Technical Committee on Data Engineering*, 21(1), 1998.

[61] B. H. Murray and A. Moore. Sizing the internet. A White page, Cyveillance, July 2000.

[62] Miki Nakagawa and Bamshad Mobasher. A hybrid web personalization model based on site connectivity. In *Fifth WebKDD Workshop*, pages 59–70, 2003.

[63] O. Nasraoui, H. Frigui, A. Joshi, and R. Krishnapuram. Mining web access logs using relational competitive fuzzy clustering. In *The Eighth International Fuzzy Systems Association World Congress*, 1999.

[64] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *The 20th VLDB Conference*, 1994.

[65] D. M. Nichols. Implicit rating and filtering. In *The Fifth DELOS Workshop on Filtering and Collaborative Filtering*, 1997.

[66] G. Paliouras, C. Papatheodorou, V. Karkaletsis, and C. D. Spyropoulos. Clustering the users of large web sites into communities. In *the 2000 International Conference on Machine Learning*, 2000.

[67] P. Pantel and D. Lin. Document clustering with committees. In *The 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

[68] Miranda Lee Pao. Concepts of information retrieval, 1989. Englewood, CO: Libraries Unlimited, Inc.

[69] J.-S. Park, M.-S. Chen, and P.S. Yu. Using a hash-based method with transaction trimming for mining association rules. *IEEE Transactions on Knowledge and Data Engineering*, 9(5):813–825, 1997.

[70] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites, 1997. Machine Learning, 27, 313-331.

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

[72] J. Pei, J. Han, and L. V. S. Lakshmanan. Mining frequent itemsets with convertible constraints. In *The 2001 International Conference on Data Engineering (ICDE '01)*, 2001.

[73] Mike Perkowitz and Oren Etzioni. Adaptive web sites: Automatically synthesizing web pages. In *AAAI/IAAI*, pages 727–732, 1998.

[74] Mike Perkowitz and Oren Etzioni. Towards adaptive web sites: Conceptual framework and case study. *Artificial Intelligence*, 118(5):245–275, 2000.

[75] PersonaLogic. URL: http://www.personalogic.com.

[76] G. Piatetsky-Shapiro, U. M. Fayyad, and P. Smyth. From data mining to knowledge discovery: An overview, 1996. In U.M. Fayyad, et al. (eds.), Advances in Knowledge Discovery and Data Mining, 1-35. AAAI/MIT Press.

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

[78] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews, 1994. Proceedings of the 1994 Computer Supported Collaborative Work Conference.

[79] P. Resnick and H. R. Varian. Recommender systems, 1997. Communications of the ACM, 40(3), 56-58.

[80] M. Rosenstein and C. Lochbaum. Recommending from content: Preliminary results from an e-commerce experiment. In *Conference on Human Factors in Computing*, 2000.

[81] G. Salton. Automatic text processing, 1989. Addison-Wesley.

[82] Gerard Salton and M. J. McGill. Introduction to modern information retrieval, 1983. McGraw-Hill, New York.

[83] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental svd-based algorithms for highly scaleable recommender systems. In *The Fifth International Conference on Computer and Information Technology*, 2002.

[84] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining algorithm for mining association rules in large databases. In *The 21st Very Large Data Base Conference (VLDB-95)*, 1995.

[85] I. Schwab, A. Kobsa, and I. Koychev. Learning user interests through positive examples using content analysis and collaborative filtering, 2001. User-Modeling and User-Adapted Interaction.

[86] Cyrus Shahabi, Amir M. Zarkesh, Jafar Adibi, and Vishal Shah. Knowledge discovery from users web-page navigation. In *Workshop on Research Issues in Data Engineering*, 1997.

[87] Upendra Shardanand and Patti Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.

[88] H. Shimazu. Expertclerk: Navigating shoppers' buying process with the combination of asking and proposing. In *The Seventeenth International Joint Conference on Artificial Intelligence*, 2001.

[89] B. Smyth and P. Cotter. A personalized tv listings service for the digital tv age. *Knowledge-based System*, 13:53–59, 2000.

[90] E. Spertus. Parasite: Mining structual information on the web. *Computer Networks and ISDN Systems*, 29(8-13):1205–1215, 1997.

[91] Myra Spiliopoulou, Jaideep Srivastava, Ron Kohavi, and Brij M. Masand (editors). Workshop on web mining for e-commerce (webkdd '00), 2001.

[92] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2):12–23, 2000.

[93] G. Strang. Linear algebra and its applications, 1988. Harcourt Brace, New York.

[94] Ah-Hwee Tan. Text mining: The state of the art and the challenges. In *PAKDD'99 Workshop on Knowledge discovery from Advanced Databases (KDAD'99)*, 1999.

[95] T. Tran and R. Cohen. Hybrid recommender systems for electronic commerce, 2000. AAAI Workshop on Knowledge-based Electronic Markets.

[96] C. van Rijsbergen. Information retrieval, 1979. Butterworth, London.

[97] K. Wang, Y. He, and J. Han. Mining frequent itemsets using support constraints. In *The 26th Very Large Data Base Conference (VLDB-00)*, 2000.

[98] W. Wong and A. Fu. Incremental document clustering for web page classification, 2000. IEEE 2000 Internatial Confernece on Information Society in the 21st century: emerging technologies and new challenges (IS2000), Nov 5-8, 2000, Japan.

[99] T. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal. From user access patterns to dynamic hypertext linking. In *The 5th International World Wide Web Conference*, 1996.

[100] Arbee L.P. Chen Yi-Hung Wu, Yong-Chuan Chen. Enabling personalized recommendation on the web based on user interests and behaviors. In *11th International Workshop on Research Issues in Data Engineering*, 2001.

[101] Kai Yu, Xiaowei Xu, Martin Ester, and Hans-Peter Kriegel. Selecting relevant instances for efficient accurate collaborative filtering. In *The 10th CIKM Conference*, 2001.

[102] Osmar R. Zaïane. Resource and knowledge discovery from the internet and multimedia repositories, 1999. Ph.D. Thesis, School of Computing Science, Simon Fraser University.

[103] Osmar R. Zaïane, Mohammad El-Hajj, and Paul Lu. Fast parallel association rule mining without candidacy generation. In *the IEEE 2001 International Conference on Data Mining (ICDM '2001)*, 2001.

[104] Osmar R. Zaïane, Jaideep Srivastava, Myra Spiliopoulou, and Brij M. Masand (editors). Workshop on web mining for usage patterns & profiles (webkdd '02), 2003.

[105] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *The 21st International Conference on Research and Development of Information Retrieval*, 1998.

[106] Amir Zarkesh, Jafar Adibi, Cyrus Shahabi, Reza Sadri, and Vishal Shah. Analysis and design of server informative www site. In *The Sixth International Conference on Information and Knowledge Management*, 1997.

[107] Tong Zheng. Webframe: in pursuit of computationally and cognitively efficient web mining, 2004. Ph.D. Thesis, Department of Computing Science, University of Alberta.

[108] Tingshao Zhu, Russ Greiner, and Gerald Haeubl. An effective complete-web recommender system. In *The Twelfth International World Wide Web Conference(WWW2003)*, 2003.

[109] Tingshao Zhu, Russ Greiner, and Gerald Haubl. Learning a model of a web user's interests. In *The 9th International Conference on User Modeling*, 2003.