University of Alberta

WebFrame: in Pursuit of Computationally and Cognitively Efficient Web Mining

by

Tong Zheng 

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

Department of Computing Science

Edmonton, Alberta
Spring 2004

Your file Votre référence
ISBN: 0-612-96344-6
Our file Notre référence
ISBN: 0-612-96344-6

"Human life is limited, but knowledge is limitless." — Zhuang, Zi

To my loving parents Yuxia, Zerong
and my dear wife Hui

Abstract

The idea of web mining is to apply the tools and techniques of data mining to the world wide web data to induce "interesting" consequences that can be used to improve various web applications.

The goal of web mining is relatively simple: provide both computationally *and* cognitively efficient methods for improving the value of information to users of the WWW. The need for computational efficiency is well-recognized by the data mining community, which sprung from the database community concern for efficient manipulation of large datasets. The motivation for cognitive efficiency is more elusive but at least as important. In as much as cognitive efficiency can be informally construed as ease of understanding, what is important is any tool or technique that presents cognitively manageable abstractions of large datasets.

Of note is the observation that no application of any learning method to web data makes sense without first formulating a goal framework against which that method can be evaluated. This simple idea is typically the missing ingredient of many WWW mining techniques. Like many data mining methods, the weakest ingredient is the formulation of discovery goals. These goals are vital to the development of evaluation criteria for any learning method.

We present our initial development of a framework for gathering, analyzing, and redeploying web data. Similar to conventional data mining, the general idea is that good use of web data first requires the careful selection of data (both usage and content data), the deployment of appropriate learning methods, and the evaluation of the results of applying the results of learning in a web application. While we use web abstraction to refer to any certain abstracted form of a particular web space

(including web content, web structure, and web usage), our framework includes tools for building, using, and visualizing such web abstractions. Our development of this framework is itself an experiment, based on our belief that we need such a framework to assess the various combinations of data, learning, and application evaluation methods.

We present an example of the deployment of our framework to navigation improvement. We focus on the idea of web usage mining and the application of simple learning methods to improve user navigation. The abstractions we develop are called Navigation Compression Models (NCMs), and we show a method for creating them, using them, and visualizing them to aid in their understanding. Also, we present a general class of methods for evaluating these intended navigation improvements, and describe the experiments that we have conducted as the basis for building insight into the navigation improvement problem and the general problem of web mining.

We hope to incrementally elaborate our framework so that we can eventually gain insight into such questions as "How can mining goals be formulated to provide learning method evaluation criteria?" or "What are the trade-offs between intrusive data gathering and navigation improvement?"

Acknowledgements

I would like to thank many people for making this dissertation possible. The first person I would like to thank is my supervisor, Dr. Randy Goebel. During all these years I have known Randy not only as an excellent supervisor, but also as a good friend. I would like to thank him for his confidence in my capability and for guiding me through the research with enthusiasm, patience, and great advice and insights.

I would like to thank my committee members, Dr. Russell Greiner, Dr. Osmar R. Zaïane, Dr. Raymond T. Ng, and Dr. Donald Heth, for carefully reading my thesis and providing many valuable comments and suggestions. I would like to thank Dr. Dekang Lin and Patrick Pantel for providing the committee-based document clustering program used in our experiments. I would also like to thank my colleagues, Tingshao Zhu and Yonghe Niu, for their help in this research.

Thanks to my parents, Yuxia and Zerong, for their love and support in all these years. Thanks to my beloved wife, Hui, for her love and patience during the Ph.D. period, and for all the happiness she has brought to my life.

Finally, I would like to thank all my friends, especially Xinguang Chen, Guohu Huang, Jiankang Wang, and Kui Wu, for their friendship and the time we have been together.

This research has been supported and funded by the Canadian Natural Sciences and Engineering Research Council (NSERC), and by the Institute for Robotics and Intelligent Systems (IRIS) Networks of Centers of Excellence, and by the Alberta Ingenuity Centre for Machine Learning (AICML). I would like to thank them for all their support. I am also grateful for the Department of Computing Science at the University of Alberta for providing an excellent work environment during the past years.

Contents

1 Introduction 1
- 1.1 Problem and Motivation . . . 1
- 1.2 Web Mining . . . 3
- 1.3 Web Mining Framework . . . 5
- 1.4 Objectives and Contributions of this Research . . . 6
- 1.5 Overview of the Dissertation . . . 7

2 Background 9
- 2.1 Web Content Mining . . . 9
  - 2.1.1 All-purpose Search . . . 9
  - 2.1.2 Special-purpose Search . . . 11
  - 2.1.3 Page Categorization . . . 12
  - 2.1.4 Page Clustering . . . 13
- 2.2 Web Usage Mining . . . 14
  - 2.2.1 Data Preparation . . . 14
  - 2.2.2 Learning Methods . . . 18
    - 2.2.2.1 Association Rule Mining . . . 18
    - 2.2.2.2 Sequential Pattern Mining . . . 19
    - 2.2.2.3 Clustering . . . 20
    - 2.2.2.4 Classification . . . 23
    - 2.2.2.5 Markov Models . . . 24
    - 2.2.2.6 Specialized Learning Methods . . . 24
  - 2.2.3 Applications of Web Usage Mining . . . 24
    - 2.2.3.1 Navigation Optimization . . . 25
    - 2.2.3.2 Search Optimization . . . 26
    - 2.2.3.3 Caching and Throughput Optimization . . . 26
    - 2.2.3.4 User Demographics . . . 27
    - 2.2.3.5 Summary . . . 27

3 WebFrame — a General Framework for Web Mining 29
- 3.1 A Caricature of a Web Mining Problem Instance . . . 29
- 3.2 System Architecture — WebFrame . . . 30
- 3.3 Evaluation of Performance Improvement . . . 31
- 3.4 Visualization . . . 33
  - 3.4.1 Web Site Structure Visualization . . . 34
  - 3.4.2 Web Data Visualization . . . 35

4 Experiments with Navigation Optimization — a Web Usage Mining Instance **37**
4.1 Navigation Optimization — a Web Usage Mining Instance 37
4.2 Navigation Compression Models 37
4.2.1 NCMs for Dynamic Recommendation 38
4.2.2 NCMs in an Unique Form? 39
4.3 Experiment Methodology 40

5 Data Selection and Preparation **42**
5.1 Data Selection 42
5.2 Session Identification 43
5.2.1 User Identification 43
5.2.2 Session Boundary Determination 45
5.2.3 Page Cleaning 45
5.2.4 Path Completion 46
5.2.5 Page View Processing 46
5.2.6 Experiments 48
5.2.6.1 Experiment 1: User Identification Methods 48
5.2.6.2 Experiment 2: Session Identification Methods 48
5.3 Content Page Identification 52
5.3.1 Content Page Identification Methods 52
5.3.1.1 Reference Length (RL) 52
5.3.1.2 Maximal Forward Reference (MFR) 58
5.3.1.3 MFR-RL 58
5.3.2 Significance of Content Pages 59
5.3.3 Accuracy of Heuristically Identified Content Pages 59

6 Learning **61**
6.1 Association Rule Mining 61
6.1.1 Interestingness of Association Rules 62
6.1.2 Significance of Association Rules 64
6.1.3 Matching Order of Association Rules 65
6.1.4 Coverage of Association Rules 65
6.1.5 Redundancy of Association Rules 65
6.1.6 Content-Page-Related Association Rules 66
6.2 Sequential Rule Mining 67
6.3 Clustering 68
6.3.1 Usage-based Clustering 68
6.3.1.1 "Interesting" Page Clusters 69
6.3.1.2 Significance of Page Clusters 70
6.3.1.3 Matching Order of Page Clusters 70
6.3.1.4 Coverage of Page Clusters 70
6.3.1.5 Content-Page-Related Page Clusters 70
6.3.2 Content-based Clustering 70
6.3.2.1 CBC Algorithm 72
6.3.2.2 Matching of Usage-based Clusters and Content-based Clusters 73

7 Evaluation 74
7.1 Evaluation Function . . . 74
7.2 Navigation Compression Mechanism . . . 76
7.3 Experiments . . . 78
7.3.1 Training Data and Test Data . . . 78
7.3.2 Experiment Plan . . . 79
7.3.2.1 Content Page Determination . . . 79
7.3.2.2 Learning . . . 81
7.3.2.3 Navigation Compression . . . 82
7.3.2.4 Evaluation . . . 82
7.4 Experimental Results . . . 82
7.4.1 Association Rules . . . 82
7.4.1.1 Experiment 1: Support and Confidence Settings . . 83
7.4.1.2 Experiment 2: Maximum Confidence Threshold . . . 88
7.4.1.3 Experiment 3: "Interesting" Association Rules . . . 88
7.4.1.4 Experiment 4: Varying Number of Association Rules 90
7.4.1.5 Experiment 5: Varying Number of Content Pages . 92
7.4.1.6 Experiment 6: Time Effect . . . 93
7.4.1.7 Experiment 7: More Training and Test Data 94
7.4.1.8 Summary . . . 94
7.4.2 Sequential Rules . . . 97
7.4.3 Combined Rules . . . 97
7.4.4 Usage-based Clusters . . . 98
7.4.4.1 Experiment 1: Similarity and Support Settings . . . 98
7.4.4.2 Experiment 2: Maximum Similarity Threshold . . . 103
7.4.4.3 Experiment 3: "Interesting" Page Clusters 104
7.4.4.4 Experiment 4: Varying Number of Page Clusters . . 104
7.4.4.5 Experiment 5: More Training and Test Data 106
7.4.4.6 Summary . . . 106
7.4.5 Content-based Clusters . . . 109
7.5 More Data: The "Music Machines" Web Logs . . . 110
7.6 Annotated Data: The "Travel Study" Web Logs . . . 112
7.7 Examples of WebKIV Visualizations . . . 115
7.7.1 Visualization of Web Site Structure . . . 115
7.7.2 Visualization of Web Surfing Animation . . . 116
7.7.3 Visualization of Web Page Usage vs. Hyperlink Usage 118
7.7.4 Visualization of NCM Application . . . 118

8 A NCM-Based Recommendation System 120
8.1 Introduction . . . 120
8.2 System Architecture . . . 120
8.2.1 User Tracking . . . 122
8.2.2 Session Management . . . 122
8.2.3 Footer Generation . . . 122
8.2.4 Recommendation Generation . . . 123
8.2.5 Feedback Management . . . 123
8.3 Summary . . . 124

9 Conclusions and Future Work **125**
9.1 Conclusions 125
9.2 Future Work 127

Bibliography **128**

List of Figures

3.1 Basic Web Mining Architecture 31
3.2 System Architecture — WebFrame 32
3.3 A Screen Shot of WebKIV — Our Web Visualization Tool 35

4.1 Using NCMs for Dynamic Recommendations 38
4.2 Experiment Methodology 41

5.1 UACS Access Log Example — One Log Record 43
5.2 Experimental Results for User Identification Methods 50
5.3 Experimental Results for Session Identification Methods 53
5.4 Exponential Q-Q Plot of Viewing Time (10/2002) 54
5.5 Viewing Time Distribution (10/2002) 55
5.6 Experimental Results for Cutoff Viewing Time Calculation 57

6.1 Association Rule Examples from UACS Log Data (10/2002) 62

7.1 Navigation Compression Example 78
7.2 Experiment Plan and Parameter Control 80
7.3 Navigation Improvement vs. #Association Rules 91
7.4 Navigation Improvement vs. #Clusters 105
7.5 Visualization of Music Machines Web Site Structure 116
7.6 Visualization of Web Surfing Animation 117
7.7 Visualization of Web Page Usage vs. Hyperlink Usage 118
7.8 Visualization of NCM Application 119

8.1 NCM-based Recommendation System Architecture 121
8.2 Dynamic Recommendation View Example 123

List of Tables

2.1 Apache HTTP Server Version 2.0 — Combined Log Format 16
2.2 Example of Transaction Identification 17

4.1 Example of Different NCMs and Associated Prediction Schemes . . . 39

5.1 Experimental Results for User Identification Methods 49
5.2 Experimental Results for Session Identification Methods 51
5.3 Experimental Results for Cutoff Viewing Time Calculation 56
5.4 Refinement of Content Page Set (10/2002) 60

6.1 Content-Page-Based Classification of Association Rules 67
6.2 Content-Page-Based Classification of Page Clusters 71

7.1 Number of Content Pages obtained with RL, MFR, and MFR-RL . . 81
7.2 Experimental Results for Association-Rule-based NCMs (AR-1.1) . . 84
7.3 Experimental Results for Association-Rule-based NCMs (AR-1.2) . . 85
7.4 Experimental Results for Association-Rule-based NCMs (AR-1.3) . . 86
7.5 Experimental Results for Maximum Confidence Threshold 89
7.6 Experimental Results for "Interesting" Association Rules 89
7.7 Experimental Results for Varying Number of Association Rules . . . 90
7.8 Experimental Results for Varying Number of Content Pages 92
7.9 Experimental Results for Weekly Updated Knowledge 94
7.10 Experimental Results on More UACS Log Data 95
7.11 Experimental Results on More UACS Log Data (Continued) 96
7.12 Experimental Results for Usage-Cluster-based NCMs (UC-1.1) . . . 99
7.13 Experimental Results for Usage-Cluster-based NCMs (UC-1.2) . . . 100
7.14 Experimental Results for Usage-Cluster-based NCMs (UC-1.3) . . . 101
7.15 Experimental Results for Maximum Similarity Threshold 103
7.16 Experimental Results for "Interesting" Page Clusters 104
7.17 Experimental Results for Varying Number of Page Clusters 106
7.18 Experimental Results on More UACS Log Data 107
7.19 Experimental Results on More UACS Log Data (Continued) 108
7.20 Experimental Results for Content-based Page Clusters 111
7.21 Experimental Results on Music Machines Log Data 113
7.22 Verification of Content Page Identification Methods 114

Chapter 1

Introduction

1.1 Problem and Motivation

The expeditious development of the Internet has dramatically extended the amount of information that can be gathered in people's everyday lives. Nowadays the World Wide Web has become a gigantic, globally distributed information center for news, entertainment, advertisements, e-commerce, government, education, and a lot of other information resources. Because the information space available to the Web users is enormous, there arises the increasing demand for efficient resource and knowledge discovery. However, the Web possesses the following intrinsic characteristics which make this task challenging:

- The Web is extremely huge. Measuring the size of the entire Web is a mission impossible to fulfill. However, several experiments have been conducted to obtain some rough estimations. According to Bharat and Broder's work [3, 4], by March 1998 there were at least 275 million distinct, static pages throughout the Web, and this number continued to grow at approximately 20 million pages per month. A later study by Murray and Moore [48] showed that by July 2000 there were more than two billion unique, publicly accessible pages on the Web, and the estimated growth rate was about 7.3 million pages per day.
- The Web is highly unorganized. While a small part of the Web (e.g., web search portals, company web sites, university web sites, or personal web sites) could be well structured and maintained, the Web as a whole is highly unstructured. The Web does not have a root node, and web resources are distributed in an uncontrolled way. There is no standard mechanism to guarantee the finding of all existing web resources.
- The web data "changes" rapidly. While the Web grows fast in size, the information it contains is also updated constantly. According to [34], by late 1996 the average lifetime of a web document was 75 days, and about 600GB of HTTP-based web data changed every month. A web page or web site may be removed or moved to a new URL. Even though the location of the web page or web site within the hyperlink structure has not changed, their content may be updated over time. Moreover, dynamic pages are becoming more and more popular nowadays, which makes the Web even more active and harder to search and analyze.

- The web data includes almost all kinds of data formats. It is currently impossible to search and analyze all formats of web data in a common way. Most efforts concentrate on HTML[1] or TXT documents, not only because users are interested in these document types in most cases, but also because these documents contain textual information which can be analyzed appropriately with widely-used, relatively mature techniques. Some data formats can be analyzed by being converted into plain text, e.g., postscript documents [43, 44]. Other data formats can be searched and analyzed as well, such as multimedia data [25, 73, 80]. But in those cases, the information retrieval task is more challenging.
- There exist a large variety of user communities throughout the Web. Users in different communities may have different backgrounds, interests, and preferences. Moreover, it is often the case that a particular user community is only interested in a very small portion of the Web. How to determine the proper relevance of web pages based on different user communities is also a challenging issue.

Due to these web characteristics, web users suffer from unguided and inefficient web usage all the time. They are often and easily lost in the web space while surfing the Web. They repeatedly find that the searching results are far from their expectations.

How to improve the efficiency of humans as web users is the problem we want to explore in this research.

Web Users & Usage

In addition to the characteristics of the Web, we also need to understand the characteristics of web users and web usage in order to improve user efficiency on the Web. A number of experiments have been conducted to characterize web user and web usage [7, 75, 74, 18]. The most important observations include:

- Revisitation refers to the user navigation action of returning to previously viewed pages. According to these experiments, revisitation is very prevalent in user navigation on the Web. Moreover, the major contribution to revisitation is provided by the few pages most recently visited. [75, 74] introduced an interesting concept — *recurrence rate*, which is defined as the probability that any URL visited is a revisit of a previous one. The recurrence rate was 61% in a three-week study during August 1994 [7], 58% in a six-week study from late October to early December 1995 [75, 74], and increased to approximately 81% in a four-month study from early October 1999 to late January 2000 [18]. Therefore, it is very important for browsing interfaces to efficiently support revisitation. Note that this recurrence rate is only well defined for static pages and accurate histories, e.g., news on the CBC webpage[2] is being updated

[1] XHTML (Extensible HyperText Markup Language) is the next generation of HTML. According to the definition from W3C (World Wide Web Consortium), XHTML is "a family of current and future document types and modules that reproduce, subset, and extend HTML, reformulated in XML" [77]. XML stands for Extensible Markup Language. As defined by W3C, XML is "the universal format for structured documents and data on the Web" [76]. Simply said, XHTML is an application of XML that represents HTML.

[2] `http://www.cbc.ca/`

constantly. A limitation of these studies is that the subjects are merely from Computer Science departments, who are clearly not representative of the entire web user population.

- Backtracking occurs very frequently. Except for following hyperlinks, the "Back" command is the most dominant action in user navigation. The "Back" command accounted for approximately 41% of all navigation actions in [7], and this rate was 30% in [75, 74]. This also explains why the major contribution to the revisitation is provided by the few pages most recently visited.
- As mentioned in [75, 74], there is a high usage of the *hub-and-spoke* structure. People often visit an index page (hub), and navigate back and forth between this index page and the pages it links to (spoke). This also explains why the "Back" command is heavily used.
- According to [18], some users tend to maintain large bookmark pages or home-pages and use these pages as indices to important or interesting web resources, while other uses prefer a compact bookmark set. In the latter case, users often repeat the same trail in order to arrive at a specific destination point, instead of using a bookmark. This usually takes a few clicks, but does not take too long. When this happens, it will be very helpful if we can predict possible destinations as early as possible and make recommendations to the user.

From the above results, it is obvious that supporting effective revisitation and backtracking is very important for web browsers. The "Back" command, bookmark mechanisms, and history mechanisms are all employed for this purpose. Moreover, many other techniques have also been studied in order to improve user navigation on the Web, such as site map, Footprints [79], Padprints [33], and a variety of other techniques.

A common characteristic of the above approaches is that they all attempt to assist user navigation merely with simple and straightforward employment of original web data, e.g., simple recording and viewing of user navigation paths and visualization of web site structure. They do not have the ability to discover hidden knowledge (or patterns) inside the data. Such being the case, web mining was introduced for its ability to reveal hidden knowledge inside the web data, including web content data, web usage data, and web structure data.

1.2 Web Mining

While more and more research is being conducted on a variety of web applications, the world wide web (WWW) has become an information playground where every possible learning technique is of potential value for improving web usage — if only one could match application performance goals with appropriate learning technologies. These learning technologies could be as simple as basic statistics, or as complex as highly complicated probabilistic prediction. Given the extremely huge size and the fast growth rate of the Web, it is a practical tautology that we can't find value without creating relevant human-oriented abstractions. Under these circumstances, web mining has gradually become a popular knowledge-discovery technology due to its capability of discovering useful information, in a finite amount of time, from a complex information space — the Web.

Simply said, web mining means data mining on the Web. However, this is a very broad definition. To be more specific, *the idea of* ***web mining*** *is to apply the tools and techniques of data mining to world wide web data to induce "interesting" consequences that can be used to improve various web applications.*

The process of web mining is to create abstractions, with the overall goal of providing both computationally *and* cognitively efficient methods for improving the value of information for WWW users. The need for computational efficiency is well-recognized by the data mining community, which sprung from the database community concern for efficient manipulation of large datasets.

In as much as cognitive efficiency can be informally construed as ease of understanding, what is important is any tool or technique that presents cognitively manageable abstractions of large datasets. The visualization of web space is based on exactly this idea: that a certain abstracted form of a large data set can provide insight into some important attributes of that space (e.g., see [35]).

So the only realistic research direction is to develop web mining software architectures that explicitly address both aspects: computational efficiency in order to provide access to large volumes of data, and cognitive efficiency in enabling users to guide the learning processes to information abstractions of appropriate relevance.

The most common instance of this combination is the application of learning to user-generated web usage data, which is referred to as *web usage mining.* In this research we use web usage mining as a specific instance of a web mining task, to illustrate the development of a general framework for web mining.

The biggest challenge is to provide a "mining" software architecture that not only provides a harness for deploying learning methods, but also aids in the incremental formulation of user mining goals. This is important because humans are the ultimate determiner of what "relevance" means.

Everyone has their idea of what an abstraction should be. For example, web search engines are dynamically created operational agents that continually update web abstractions — indices, which are then coupled with user query systems to identify relevant web information. Similarly, meta search engines provide another level of abstraction, working at a granularity above search engines by transforming single user queries into several queries to regular search engines. In the other direction, corporate, intranet, and e-business search engines provide local indexing structure, imposing more rigid abstractions that are targeted to circumscribe corporate policies, workflow constraints, and sales strategies.

This is why the notion of *web data* is as broad as the potential applications of its mining. Most current applications of web mining (e.g., [21, 71, 47, 56, 65, 85, 81]) have concentrated on data that is created by the browsing user, beginning with the usage logs produced by web servers (e.g., [72]). Of course there is a broad spectrum of such user "web data," including ordinary web logs, cookies, page exit surveys, search query collections, and even hand-collected user surveys. Even though this spectrum of information can itself be incredibly broad, there is another aspect of web data that is even broader and deeper: the web content itself.

Classifying Web Mining Methods

Web mining can be categorized into three broad areas: one is *web content mining,* which is the process of using data mining techniques to retrieve useful information

and knowledge from resources across the entire Web or inside a particular web space (e.g., a web site); another one is *web usage mining*, which is the process of discovering and analyzing user access patterns from various web usage logs (e.g., access logs, query logs, etc.), and then exploit the user access patterns we have discovered to various web applications in order to improve their performance; the last one is *web structure mining*, which is the process of obtaining and using link structures of the web graph.

In this research we focus on web usage mining, while a number of techniques for web content mining and web structure mining are also briefly discussed. Within this general pursuit, there are at least four broad categories of possible applications, including navigation optimization, search optimization, caching and throughput optimization, and the development and use of user demographic models. For each process (navigation, search, caching and throughput, and user demographics), we require measures that can provide a basis for determining the value of the web usage mining we undertake.

As a specific example, we explore the navigation optimization problem, trying to improve user navigation in a web site by creating and using a special form of abstractions — *Navigation Compression Models.*

1.3 Web Mining Framework

As noted earlier, web mining is the process of applying data mining techniques to world wide web data to obtain valuable knowledge that can be used to improve various web applications. While different web mining tasks may involve different kinds of applications, data sets, and learning methods, it is realized that all the web mining processes do share the same basic phases: (1) data preparation, (2) learning, and (3) evaluation.

Data preparation is the process of collecting and preprocessing web data for learning algorithms. In a preliminary development, data preparation can be as simple as text extraction from web log files. In the spectrum from logging HyperText Transfer Protocol (HTTP) requests to conducting user surveys, the data preparation process can be as sophisticated as complete natural language analysis of user query logs. Similarly, applicable learning methods range from simple association rule induction (e.g., correlating navigation paths to associated pages within sites), to sophisticated hypothesis formation based on time use and page content analysis. The third and final phase is application driven, and requires the specification of measures that can be applied before and after learning, in order to provide a basis for assessing the value of various data and learning combinations.

Previous work on web mining systems can be found in [21, 71, 47, 81]. The WEBMINER system described in [21] and the WebSIFT system described in [71] are two prototypes of web usage mining systems, in which the general web usage mining process is divided into three major phases: data preprocessing, knowledge discovery, and pattern analysis. [47] presents a specialized web usage mining system — usage-based web personalization system, which learns aggregate user behavior patterns from web usage logs, and makes recommendations to individual users based on the obtained knowledge. In this system the web personalization process is also divided into three phases: data preparation, usage mining, and recommendation.

[81] describes the design of a database-driven web log mining system, WebLogMiner, which performs data mining on web server log files using data cube and On-line Analytical Processing (OLAP) techniques [14, 31].

The aforementioned web mining systems all involve the data preparation phase and learning phase. Moreover, the discovered knowledge can be analyzed and inquired [21, 71], or applied to aid in certain web applications, e.g., improving user navigation with dynamic recommendations [47]. However, there is a common missing ingredient in these systems — the evaluation phase. In this research, we will explore the difficulty of the evaluation problem, and present our first effort in addressing this problem.

The Web Mining Framework proposed in this research is designed to facilitate all aspects of web mining we can currently envisage, including the use of browsing data, web content, and web meta content. Our development of this framework is itself an experiment, based on our belief that we need such a framework to assess the various combinations of data, learning, and application evaluation methods.

In what follows, we will provide a detailed description of the current status of our framework, together with examples of our preliminary experiments. In all cases we attempt to be as general as possible in identifying the "inductive opportunities" that arise within web data, and anticipate their ultimate role in improving the value of a range of web activities.

1.4 Objectives and Contributions of this Research

We present our initial development of a web mining framework for gathering, analyzing, and redeploying web data. Currently there exist much theoretical and experimental work on web mining systems and techniques, but they all reveal some sort of weakness, e.g., non-comprehensive architecture of an integrated web mining system, incomplete comparison between various data preparation and learning methods, and finally, the lack of application evaluation methods. In this research we make an effort to overcome these deficiencies in the framework we propose. Our short term goal is to build a prototype of such a framework, and gain some insight into it with an example of the deployment of this framework to the navigation compression problem. For a long term objective, this framework can be used as a basis to build more sophisticated web mining systems, or serve as a test-bed to examine the effects of various web mining techniques.

The general architecture of our framework is more comprehensive than that of any other existing web mining systems. The application evaluation module, which is missing in most existing web mining systems, is included in our framework as an important final phase. Moreover, an independent visualization module is combined into our framework to aid in the human interpretation of the data and knowledge across the web mining process.

As an example, our framework is deployed to address the navigation compression problem. While the goal of navigation compression is to help users reach the contents of interest more quickly, our basic idea is to learn from previous user behaviors as well as the content and structure in a web site, and then use the discovered knowledge to improve user navigation by dynamic recommendations. In the experiments, a number of combinations of data preparation, learning, and application evaluation

methods are explored and compared with each other.

While we use web abstraction to refer to any certain abstracted form of a particular web space (including web content, web structure, and web usage), in the learning process we create an operable web abstraction — Navigation Compression Models (NCMs), which are functions that predict user destination points based on the user's current traversal path. In this process we focus on the idea of mining "interesting" traversal patterns. Similarly, in the recommendation process, we focus on the idea of suggesting "interesting" pages. In the final phase of our framework, we propose a variety of application evaluation methods based on the number of traversed links.

Through this research we expect to obtain some evidence to support our hypothesis that we can learn from various web data in order to improve the performance of certain web applications. We hope to incrementally improve our framework, and develop answers to questions like:

- "How can mining goals be formulated to provide learning method evaluation criteria?"
- "What kind of knowledge can we expect from the data for the purpose of improving user navigation?"
- "What data can be used to obtain the knowledge we expect?" and "Is the data sufficient or is additional data required?"
- "Is the data appropriate to apply learning methods to, or is preprocessing required?" and "Is the data clearly self-explained, or assumptions have to be made?"
- "How well can we improve the performance of the application?" and "Can we do better with more data?"
- "What are the tradeoffs between intrusive data gathering and navigation improvement?"

1.5 Overview of the Dissertation

This dissertation is organized as follows. Chapter two introduces the background of this research, including a number of major existing web mining techniques and applications. As noted earlier, in this research we focus on web usage mining. Therefore, the techniques and applications in web usage mining are discussed in detail, while those in web content mining are only discussed briefly. Web structure mining is not discussed separately because it is generally combined in the processes of web content mining and web usage mining. Chapter three presents the general framework we propose for web mining, called WebFrame. The system architecture and major components of the framework are described in detail. Moreover, the major contributions of our framework are also discussed, including the evaluation of performance improvement and the visualization module. Chapter four proposes our experiments with an exemplary application of our framework, which is the application of improving user navigation by navigation compression. The navigation compression problem is defined, as well as our approach to this problem — the creation and use of Navigation Compression Models. The experimental methodology is presented, and the entire experimental procedure is divided into three parts corresponding to the three basic components of our web mining framework. Chapter five, six and seven present our experiments with the three parts, step by step. The experimental results are

evaluated based on the *Navigation Improvement* (NI) criterion. Also an example is presented showing the use of visualization. Chapter eight presents a NCM-based dynamic recommendation system we have developed. Except for the architecture, basic requirements, and interfaces of the system, we also discuss a simple attempt of gathering and using intrusive data through the user feedback mechanism. Finally, chapter nine concludes this dissertation and discusses future work.

Chapter 2

Background

2.1 Web Content Mining

The primary objective of web content mining is to use various data mining techniques to discover useful information and knowledge, related to a particular topic specified with text or other formats, from the resources across the entire Web or in a restricted web space. While the traditional information retrieval techniques focus on finding useful information with keyword extraction and textual statistics of the documents, web content mining focuses on discovering useful knowledge from the information and improving the results of information retrieval by applying various learning techniques.

In this section we present an overview of the major mining techniques involved in existing search engines. In addition, other important applications of web content mining are also discussed, such as page categorization and page clustering.

2.1.1 All-purpose Search

The traditional all-purpose search is a primary application of web content mining. As noted earlier, the Web has some intrinsic characteristics that make web searching very difficult, especially when the search space is the entire Web. Creating an effective search engine involves challenging issues in a number of different ways. First, searching the entire Web is a task that will never be completed. Therefore, we may want a crawling method with which more "important" pages can be indexed first. Second, there can be considerable volume of searching results more or less matching a given topic. Such being the case, we need a ranking scheme so that more "important" pages have higher ranks, therefore can be accessed first. Third, since the web data changes rapidly, the indices created for searching need to be updated on a regular basis, and need to be finished in a finite amount of time. Finally, using merely a keyword-based approach is not enough for effective searching and ranking, because some highly related web pages may not contain the searching keywords.

Cho et al. [17] explored the idea that more "important" or "meaningful" pages should be indexed first, and studied how should a crawler determine the importance of URLs it has observed. They proposed a set of metrics to determine the importance of a web page, as shown in the following list:

- *Text Similarity* This metric is the similarity between the textual content of

a web page P and a text query Q. It is particularly useful for building a specialized database on a particular topic, based on the assumption that pages referring to the text query are more important for the topic.

- *PageRank* [5] This is a ranking scheme used to measure the popularity of a web page. It is described in detail later in this section.
- *Backlink Count* This metric is the number of links pointing to a web page.
- *Forward Link Count* This metric is the number of links going out of a web page.
- *Location Metric* This metric indicates the importance of a page's location in the web space. For example, URLs ending with ".com" may be assumed more important (or less important, depending on the user's preference) than URLs with other endings.

The first two metrics are also the essential metrics for ranking searching results. These importance metrics can be used exclusively or combined together, depending on varied crawling purposes. For example, to search for content pages, we may prefer pages with higher text similarity, more backlinks or higher PageRank values, and maybe fewer forward links.

A lot of work has been done on the ranking of searching results. So far there are two kinds of such ranking schemes: one is *keyword-based*, which indicates how to rank documents based on text similarity; another one is *popularity-based*, which indicates how web link structure can be employed to determine the importance of web documents.

Many existing information retrieval techniques can be applied for keyword-based ranking, such as the vector space model [64]. A keyword-based rank can be computed based on the textual content and meta-information related to the document, such as the count of keyword appearance, position in document (e.g., title, anchor, URL, plain text, etc.), font size, capitalization, and keyword appearance in the anchor text on pages that link to the document, etc. [5].

There exist a number of approaches for computing popularity-based ranks. But all these approaches originated from the same hypothesis: *a web page is determined to be popular, therefore probably more "important," if it is highly referenced, or linked to, by other pages in the same topic.* So far there are two well-known algorithms for computing popularity-based ranks: one is PageRank [5]; another one is HITS (Hypertext-Induced Topic Search) [10, 27, 38, 11, 37].

PageRank measures page popularity based on three assumptions:

1. The PageRank value of a page is affected by the PageRank values of those pages pointing to it.
2. The influence of a page on those pages it has a path to, decreases as the path grows longer. In [5], this intuition is formulated as a damping factor d.
3. The more outgoing links a page has, the less influence it has on each of those pages it links to.

Suppose page A has pages $\{T_1, \cdots, T_n\}$ pointing to it, $C(A)$ is defined as the number of links going out of page A, d is a damping factor between 0 and 1 (usually set to 0.85, according to [5]), then the PageRank value of page A is computed as:

$$PR(A) = (1-d) + d\left[\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)}\right] \tag{2.1}$$

The PageRank values of all pages in a certain web space can be computed through the following spreading scheme: first, all PageRank values are initially set to a uniform positive constant, and then, the computations in Equation (2.1) are iterated (and normalized) until the result converges.

The HITS algorithm further refines the concept of popularity into *authorities* and *hubs*: "A good authority is a page that is pointed to by many good hubs, while a good hub is a page that points to many good authorities." [10, 27, 37] The authorities and hubs are computed with the following two steps [10]:

1. Obtain a *result set* through the following steps:
   (a) Collect a *root set* of pages (say, 200) from an index-based search engine, using the query terms.
   (b) Expand this root set to a *base set* by including all pages that are linked to by pages in the root set, and all pages that link to a page in the root set, up to a cutoff size.
   (c) In the base set, remove all links between pages within the same domain.
2. Suppose each page p in the result set has a non-negative authority weight x_p, and a non-negative hub weight y_p. All $x-$ and $y-$values are initially set to a uniform positive constant. Then, the following computations are iterated (and normalized) until the result converges:

$$x_p = \sum_{q \to p} y_q \ , \ y_p = \sum_{p \to q} x_q \tag{2.2}$$

   Given the link matrix A of the result set, where

$$A_{ij} = \begin{cases} 0, & \text{if there is no such link } \{\text{page } i \to \text{page } j\} \\ 1, & \text{if there exists a link } \{\text{page } i \to \text{page } j\} \end{cases} \tag{2.3}$$

   the result of $x-$ and $y-$vector will converge to the dominant eigenvector of $A^T A$ and AA^T, respectively. A similar case happens in the PageRank algorithm: the result of PageRank will converge to the dominant eigenvector of the link matrix of the restricted web space.

A problem with HITS is that it might bring the results of more general, even irrelevant topics. In such cases anchor text and the text around them can be used to get better results [10].

PageRank and HITS are two examples of using web structure mining to improve the results of information retrieval. However, it is worth noting that these two techniques have different purposes. PageRank is used to determine the general popularity of a web document based on both its inlinks and outlinks. HITS is used to determine the popularity of a web document *regarding a specific topic*, and the popularity is refined into authorities and hubs which are based on the inlinks and outlinks respectively.

2.1.2 Special-purpose Search

The aforementioned is about the mechanism of all-purpose search engines. In addition, there exist some other special-purpose search engines.

Chakrabarti et al. [8] proposed a topic-specific searching scheme, called *focused crawling*, which indexes and searches a specific set of topics based on a taxonomy tree, a "classifier" used to automatically classify documents into appropriate topic nodes in the taxonomy tree, a set of exemplary documents, and a corresponding set of starting points.

To do focused crawling, a taxonomy tree (such as that used by Yahoo![1]) and an associated pre-trained classifier should be provided as part of the initial input. The "classifier" consists of a set of statistical classification models, each of which corresponds to a topic node in the taxonomy tree. Then, a set of exemplary documents, also provided as part of the initial input, are imported in order to specify the topic set. The exemplary documents, as well as more training documents if necessary (obtained through other ways, e.g., using all-purpose search), are first classified into some topic nodes in the taxonomy tree by the automatic classifier, then adjusted by the user manually. After that, the classifier can integrate the adjustment made by the user and further refine its classification models. Moreover, the taxonomy tree can be refined by the user as well.

Finally, web exploration can be guided by relevance (determined by the classifier) and the popularity (authorities and hubs) mechanism, starting from the given set of starting points. With focused crawling, we can build distributed focused crawlers specialized for a variety of topics (*distributed search engines*), which together compose an all-purpose search engine.

Focused crawling could be much faster than traditional all-purpose searching schemes because the topics are restricted and pruning techniques can be applied while appropriate. Also this approach can be more accurate because learning techniques can be applied for training high-quality classification models. However, the traditional all-purpose searching is still necessary for providing training data and starting points for focused crawling.

Hersovici et al. [32] proposed another kind of special searching scheme, *shark-search*, which is a dynamic search based on a seed URL and a text query. Dynamic search fetches data at the time the query is made, while static search uses a pre-built index database. However, dynamic search does not scale up, therefore is more suitable for discovering information in relatively small and dynamic sub-webs. For example, it may take several minutes to search for all documents related to a specific topic in a small web site, while this searching could take much longer across the entire web space. Therefore, in the latter circumstance, a pre-built index database becomes necessary to restrict the responding time of real-time searching.

Note that it is a difficult challenge to develop a general framework to evaluate the effectiveness of various web searching schemes, and there has been much less effort in addressing this problem.

2.1.3 Page Categorization

In addition to searching, web content mining also has other important applications, e.g., page categorization. Pirolli et al. [59] proposed a technique which categorizes a web page into various page types, such as head page, content page, navigation page, reference page, etc., according to a number of features:

[1] `http://www.yahoo.com/`

- *meta-information*, e.g., file size, URL
- *topology*, e.g., inlinks, outlinks, average depth of its descendants as measured in local file system (e.g., number of '/' in URL)
- *web usage statistics*, e.g., access frequency, usage frequency as an entry point
- *text similarity analysis*, e.g., text similarity with its descendants

Chakrabarti et al. [9] proposed a term-based hierarchical text classifier, which tries to find the best content-based fitting class for a text document in a pre-defined topic taxonomy tree, using text analysis and probability theory. This technique is further refined in [12] by introducing hyperlink structure to make it more suitable for hypertext categorization. The classification method described in these two papers were also employed in [8] to train the document classifier used in focused crawling.

The difference between Pirolli's and Chakrabarti's approaches is that the previous one concentrates on the categorization of page type, while the latter one concentrates on the categorization of a page's actual content. For example, the first method can be applied to determine that a given page is a content page, while the second method can be applied to find out that the page's content is about computer networks.

2.1.4 Page Clustering

Page categorization, also called page classification, is an example of supervised learning, in which the learning of the models is "supervised" with a pre-defined class set (or class hierarchy) and a set of pre-classified training examples. Unlike page categorization, page clustering is an example of unsupervised learning, which does not rely on pre-defined classes and training examples.

Page clustering is used to discover clusters of *similar* web documents so that documents are similar within clusters and relatively dissimilar across clusters. The similarity between two documents is determined with a similarity function, also known as distance function, and the definition of the similarity function determines the "cognitive meaning" of the obtained clusters.

Purely content-based document clustering is an important method in information retrieval, which provides a valuable way of exploring the information space by organizing a large number of documents into a relatively smaller number of meaningful clusters based on the information residing in the document text. There are a number of existing clustering algorithms available for this task, e.g., K-means [45], Buckshot [22], Chameleon [36], and CBC [53]. Moreover, in the mission of clustering web pages, the hyperlink structure is another kind of information that can be employed together with document text to achieve improved cluster quality.

Weiss et al. [78] proposed a clustering algorithm, called content-link hypertext clustering, which clusters hypertext documents based on both document contents and hyperlink structure. The content-link clustering algorithm uses a similarity function that includes a link-based component and a term-based component. The link-based component captures three important factors about hyperlink structure: the length of the shortest path between two documents, the number of their common ancestors, and the number of their common descendants. The term-based component also captures three factors: term frequency, term attribute, and a document size factor which offsets high term frequencies of terms in large documents.

2.2 Web Usage Mining

Web usage mining originated from the following hypothesis: *we can learn from web usage data in order to improve the performance of various web applications.* The predecessor of web usage mining can be traced back to earlier work on web log analysis, which performs simple statistics on web log data, such as most frequently requested documents, activity level by day of week or hour of day, average number of requests per visit, average number of visits per day, most active organizations, etc. [72]. However, researchers quickly found that there is more useful information hidden in the usage data which is, since then, widely referred to as *user access patterns.*

The primary difference between web usage mining and web content mining is the data to be analyzed. While web content mining concentrates on the web content itself, web usage mining concentrates on the usage data which shows how users access the web content.[2] In this section we present an overview of the usage data that can be analyzed, and focus on the preprocessing of the server-produced web access logs which are the most important and largest data source of web usage. After that, we present the state of the art of web usage mining with a number of existing learning techniques. Finally, the potential applications of web usage mining are discussed.

2.2.1 Data Preparation

What we call "web usage data" is the data that records user activities on the Web. There are basically two kinds of web usage data: one is the log files automatically recorded by web servers or certain software agents; another one is the user survey data collected through various inquiring techniques, such as interviews, questionnaires, etc. User survey gathering is generally demand-driven, and different demands often lead to different survey templates. The primary benefit of user survey data is its ease of use for the surveyors. However, as indicated by Cockburn and Mckenzie [18], it also has an essential limitation that it describes users' perceived, instead of actual, activities. Moreover, the acquisition of this kind of data brings a lot of intrusion, therefore is not preferred by most users. In this research we focus only on the automatically recorded usage log files.

Web usage logs can be collected at the client side or at the server side. Most existing web usage logs are collected at the server side, e.g., at HTTP web servers. The usage logs of a specific web server record all user requests within its corresponding web site. However, the logging of the server-side usage data is complicated by the existence of a number of widely-used web techniques, such as local caches, proxy servers, corporate firewalls, etc. For example, user identification is not straightforward because it is possible for different users to have the same IP address. The requests of cached (locally or at proxy servers) documents could be missed from the server logs. Moreover, the view-time estimations are inaccurate because of variations in network transfer time.

At the client side, the usage data can be collected more accurately for a number of reasons. First, at client side there is no user identification problem. Second, the cache hits missed from the server logs can be easily caught at the client side. Finally,

[2]The hyperlink structure of the Web is used in both web content mining and web usage mining processes.

the view-time estimations can be more accurate by excluding the variable network transfer time.

There exist a number of approaches to collect usage data at the client side. In [7, 75, 74, 82, 84, 83] a specially coded web browser was used to capture users' navigation behavior. In [18] users' navigation history files and bookmark files were used with the users' permission. A primary limitation of these approaches is the intrusion they bring to the users. Moreover, due to the requirement for user cooperation, the subjects are usually from restricted web communities which are not representative of the entire web user population.

Shahabi et al. [66] proposed a special kind of client-side usage data acquisition mechanism, which is implemented with a Java applet. The Java applet functions as a remote agent, which is uploaded from the web server to the client-side web browser when the user first enters the web site running this mechanism. It then captures user behaviors such as hits and view-times, and sends the usage data to a data acquisition server called "acquisitor", where the usage data is saved into database to be used for web usage mining without further preprocessing. This approach has two requirements. First, the Java applet code has to be embedded in every web page running this mechanism. Second, it requires users' cooperation to enable Java applet in the client-side web browsers.

The task of data preparation is to transform original data into an appropriate format that can be used by various learning methods. The client-side usage data is usually well prepared while being recorded. Therefore, in this section we only focus on the preprocessing of the server-produced usage data. The access logs are the most important and largest server-produced usage data source. They record information of all the interactions between users and the web server. There also exist another kind of useful usage logs — the query logs, which record users' interactions with an in-site search engine. Because the query logs are not widely available, and if available, are often combined within the access logs, in what follows we discuss only the preprocessing of the access logs.

Note that our goal in this research is to make maximal use of the non-intrusive web data and avoid intrusive data collection. Therefore, we choose to use only server access logs in the experiments because this data is widely available and non-intrusive to collect.

The server access logs can be configured to record in a variety of formats. As an example, the Apache HTTP server's combined log format is illustrated in Table 2.1.[3] The log files are in a request-based format, where each log record corresponds to a single user request. Since all the requests are logged in the order of requesting time (more accurately, the time when the server finished processing the request), requests from different users are inevitably entangled with each other.

Partitioning the log data into meaningful sequential sets is the basis for extracting useful patterns. The first necessary step is to partition the log data based on each single user. Cooley et al. [21] presented several approaches to the user identification problem. User authentication and cookies can be used to identify users quite accurately. However, these two kinds of data is not always available. In that case, this problem can be approached by combining IP address with user agent, referer,

[3]For more information about Apache HTTP server's various log formats, please refer to "`http://httpd.apache.org/docs-2.0/logs.html`".

Table 2.1: Apache HTTP Server Version 2.0 — Combined Log Format

remotehost rfc931 authuser [date] "request" status bytes "referer" "user_ agent"

Field Name	***Description***
remotehost	This is the IP address or hostname of the client (remote host) that made the request to the server.
rfc931	This is the RFC 1413 identity of the client (logname) determined by *identd* on the clients machine (set to "-" if not available).
authuser	This is the user ID of the client making the request as determined by HTTP authentication (set to "-" if not available).
[*date*]	[*dd/mmm/yyyy:hh:mm:ss <offset from GMT>*] This is the time when the server finished processing the request.
"request"	*"<method> <document> <HTTP protocol>"* This is the request line from the client.
status	This is the status code that the server sends back to the client.
bytes	This is the size of the content returned to the client, not including the response headers (set to "-" if no content was returned to the client).
"referer"	This is the URL that the client reports having been referred from.
"user_ agent"	This is the identifying information that the client browser reports about itself.

as well as heuristics related to site topology.

After identifying different users, the next step is to partition each user's request sequence into meaningful sub-sequences more appropriate for learning. There exist at least two leading definitions of such sub-sequences, *sessions* and *transactions*, corresponding to different abstraction levels. Each session includes all the page accesses from a single user during a single visit to a web site. Each transaction is a subset of a session, and can be thought of as the path a user takes to get to a content page.

In the domain of web technology, a *content page* is defined as a page that contains concrete information that the particular web site is providing. It is also believed by some researchers that content pages should be defined based on users' interest. Therefore, a page can be a content page for some users, while being useless for other users. Also, a content page can change status for the same user after some visits. Such being the case, the identification of content pages is very complex.

The biggest difficulty for determining a session is that the server generally does not know when a user leaves the web site. A page exit survey is a possible solution to this problem, but brings a lot of intrusion to the users, therefore is technically or legally infeasible under most circumstances. The simplest way, which is also the most widely used way so far, to achieve session identification, is to apply a timeout: *for a specific user, if the duration between two adjacent page requests exceeds a pre-defined timeout, it is assumed that the user is starting a new session* [21, 65]. The length of the timeout can be set as an empirical value obtained through usage statistics (e.g.,

Table 2.2: Example of Transaction Identification

Log Data

Requested Page	A	B	C	D	E	C	B	F	G
View-time (minutes)	1.0	1.5	2.5	5.0	2.0	0.2	0.5	6.0	1.2

Results of Transaction Identification

Method for Transaction Identification	Result
Reference Length (cutoff view-time = 3 minutes)	A-B-C-D, E-C-B-F, G
Maximal Forward Reference	A-B-C-D-E, B-F-G

Note: In the identification result of Maximal Forward Reference, we get "B-F-G" instead of "C-B-F-G" because "C-B" is also a backtracking.

25.5 minutes from [7]) or an arbitrary value assigned manually (e.g., 30 minutes).

Another issue that needs to be concerned is the identification of transactions. *The various methods we use to do transaction identification from the access logs can be thought of as assumptions with which content pages can be determined merely from web usage data.* Currently there exist at least two such methods that are worth mentioning: one is *reference length*, proposed by Cooley et al. [19, 21], which assumes that users generally spend more time on content pages than auxiliary pages, therefore identifies content pages based on a cutoff view-time; another one is *maximal forward reference*, proposed by Chen et al. [15], which assumes that maximal forward references are content pages, and the pages leading up to the maximal forward references are auxiliary pages. Here, a maximal forward reference is defined as the last page requested (before session timeout) by the user before backtracking occurs. An example is illustrated in Table 2.2 which clarifies the identification of transactions based on these two methods.

Due to the high diversity of the considerations of different web designers, preferences and surfing habits of different users, and the information being provided on different sites, it is obvious that identifying content pages merely from web usage data is far from accurate, and also inappropriate in terms of meaning. More reasonable approaches to this problem should take pages' actual content into account, which makes this work inevitably related to web content mining. For instance, the page categorization method proposed by Pirolli et al. [59] can be applied for content page identification. On the other hand, web usage data is not completely useless in this problem: *we believe that certain results from web usage mining can be used to further refine the categorization obtained from web content mining.* For example, if a page rarely appears as a maximal forward reference or as a page that users spend more time on, then this page's value as a content page should not be high, even if it is categorized as a content page with certain web content mining approach.

There are two other issues worth noting in the process of data preparation. One is *path completion* [21], which is the process of completing the user sessions by adding those requests not actually made to the server, therefore not appearing in the server access logs. As noted previously, requests of documents already in the local cache or proxy cache could be missed from the server logs.

Another issue is the idea of *page view*. As defined by the W3C Web Characterization Activity (WCA), a page view is the "visual rendering of a Web page in a specific client environment at a specific point in time" [40]. More specifically, a page view includes all the documents that contribute to the user's view, i.e., the display in the client-side web browser, at a particular point of time. A page view usually consists of a main page and a number of associated documents. For example, an HTML page may have several image files or other HTML files associated with it, e.g., by using the HTML elements "<img>" or "<frame>". Whenever the main page is requested, those associated documents will be requested automatically.

To track the users' behaviors accurately, it is preferable to combine all the log records of the page view into one single record — a request of the main page. However, with the current HTTP protocol, we can not always tell whether a request is made by the user or generated automatically from the page view association, by merely looking into the log files. Intuitively a simple analysis of the page content can easily solve this problem. But it brings another issue: we have to make sure the page being analyzed is *exactly* the page of the logged request. Due to the dynamic characteristic of the Web, pages are constantly being changed, moved, or deleted.

2.2.2 Learning Methods

After the process of data preparation, we can then apply various learning methods to the "meaningful" data sets in order to find useful patterns. In this section we present an overview of a number of existing learning techniques that can be applied for this purpose, which as a whole represent the state of the art of web usage mining research.

The commonly used data mining techniques that can be applied for web usage mining include association rule mining, sequential pattern mining, clustering, classification, and Markov models.

2.2.2.1 Association Rule Mining

Association rules generally capture the co-occurrence patterns of different items in individual transactions. In the domain of web log mining, association rules capture the co-occurrence relationships between different web pages based on users' navigation activities [19, 20, 21, 47]. Given A and B as sets of web pages:

$$A = \{a_1, \cdots, a_m\} \ , \ B = \{b_1, \cdots, b_n\}$$

the association rule $\{A \rightarrow B\}$ indicates that if all pages in set A are requested in a particular user transaction, it is anticipated that all pages in set B are requested in the same user transaction as well. This user transaction is a general concept representing any "meaningful" user access sequence. Therefore, it can be either a "session" or a "transaction" as defined in the previous section.

Association rules are generally created based on two criteria: *support* and *confidence*. Given an association rule $\{A \rightarrow B\}$, support indicates the possibility that A and B occur together in the same transaction, while confidence indicates the possibility that a transaction containing A also contains B. Suppose the set of user transactions is given by

$$T = \{T_1, \cdots, T_N\}$$

the set of user transactions containing A is

$$S_A = \{T_i \mid A \subseteq T_i\}$$

and the set of user transactions containing both A and B is

$$S_{AB} = \{T_i \mid A \subseteq T_i, B \subseteq T_i\}$$

then the support and confidence of the association rule are defined as follows:

$$support(A \rightarrow B) = \frac{|S_{AB}|}{|T|}$$

$$confidence(A \rightarrow B) = \frac{|S_{AB}|}{|S_A|}$$

Another important but not widely used criterion for association rule discovery is *lift*. Given an association rule $\{A \rightarrow B\}$, lift indicates how much the presence of A can improve the possibility that B appears. Therefore, an association rule with $[lift < 1]$ actually implies that the presence of A has a negative influence on the appearance of B. In addition to the above descriptions, also given the set of user transactions containing B as

$$S_B = \{T_i \mid B \subseteq T_i\}$$

then the lift is defined as

$$lift(A \rightarrow B) = \frac{confidence(A \rightarrow B)}{P(B)} = \frac{\frac{|S_{AB}|}{|S_A|}}{\frac{|S_B|}{|T|}} = \frac{|S_{AB}| \cdot |T|}{|S_A| \cdot |S_B|}$$

2.2.2.2 Sequential Pattern Mining

Similar to association rules, sequential patterns also capture the co-occurrence patterns of different items in individual transactions, only in a time-ordered fashion. In the domain of web usage mining, a sequential pattern $\{p_1, \cdots, p_n\}$ indicates a set of pages often requested as a time-ordered sequence. It is sometimes required that p_i and p_{i+1} should be adjacent in the access logs, but this is not always the case.

A variation of sequential patterns, also known as sequential rules, are also applicable to web usage mining. Sequential rules share the same format with association rules, only with time-ordered information. In a sequential rule $\{A \rightarrow B\}$, A and B are both time-ordered page access sequences, and B only appears after A. Similar to association rules, sequential rules are also discovered based on the three criteria: support, confidence, and lift.

Sequential pattern mining and sequential rule mining can be extended to inter-transaction analysis. In that case, the sequential patterns and sequential rules are not restricted to individual transactions, and the time intervals between transactions can be used to predict the time of the transaction in which a certain consequence occurs [20]. For example, in an inter-transaction sequential rule $\{A \rightarrow B\}[t]$, A and B are in different transactions of the same user, and t is the expected time interval between the two transactions.

2.2.2.3 Clustering

The idea of clustering is to group together data items with similar characteristics. In the previous section we discussed content-based page clustering in the process of web content mining. In web usage mining clustering can be applied in several different ways, including clustering of pages, clustering of user transactions, and clustering of users.

Clustering of Pages

Clusters of pages can be obtained merely from web usage data, e.g., the access logs. These clusters indicate sets of pages that tend to be requested together in the same user visit, and therefore may have related contents. Perkowitz et al. [56] proposed a graph-based algorithm, called PageGather, for purely usage-based page clustering. The idea of this algorithm differs from traditional clustering in that: traditional clustering generally attempts to partition the entire data space, while PageGather focuses on finding a relatively small number of high quality clusters. Here, high quality means the members of the same cluster are highly cohesive, i.e., the similarities between them are relatively high.

The PageGather algorithm creates an undirected and unweighted graph based on a similarity matrix computed from the co-occurrence patterns of different web pages. The similarity between two pages p_1 and p_2 is computed as:

$$sim(p_1, p_2) = \min\{P(p_2 \mid p_1), P(p_1 \mid p_2)\}$$

where $P(p_2 \mid p_1)$ and $P(p_1 \mid p_2)$ correspond to the confidence measure in association rule mining (note that this measure has to be defined over some finite scope of transactions):

$$P(p_2 \mid p_1) = confidence(p_1 \rightarrow p_2)\ ,\ P(p_1 \mid p_2) = confidence(p_2 \rightarrow p_1)$$

Moreover, this algorithm tries to discover clusters of related but *currently unlinked* pages. Therefore, if pages p_1 and p_2 are linked, then $sim(p_1, p_2) = 0$.

Each node in the graph represents a web page, and each edge indicates that the similarity between the connected two pages is greater than a given threshold. Finally, certain graph algorithms can be applied to find cliques or connected components in the graph, each of which represents a page cluster. A clique is a subgraph in which every pair of nodes have a edge between them, while a connected component is a subgraph in which every pair of nodes have a path between them. Therefore, it can be expected that connected components will result in much larger, but probably less useful clusters. Moreover, PageGather can generate overlapped clusters, which is reasonable in the web domain.

Another method that can be used for usage-based page clustering is the Association Rule Hypergraph Partitioning (ARHP) algorithm [29, 30]. Similar to PageGather, ARHP is also a graph-based algorithm. However, ARHP creates a weighted hypergraph instead of a traditional graph. A hypergraph is an extension of a graph, in which each hyperedge can connect more than two nodes.

With ARHP, each hyperedge represents a frequent item set (also called a large item set, i.e., an item set that meets a user-defined minimum support) comprised of the connected items, and the weight of the hyperedge is determined by a function of

confidences of all the association rules related to the frequent item set. For example, given a hyperedge E and its corresponding frequent item set S, suppose the set of association rules generated from S is denoted as R, and the weighting function is the average of the confidences, then the weight of hyperedge E is computed as:

$$w(E) = \frac{1}{|R|} \sum_{r \in R} confidence(r)$$

Finally, the hypergraph can be partitioned into clusters based on the hyperedge weights.

There is a cognitive problem with purely usage-based page clustering approaches: we do not know the topic (or set of topics) of each page cluster. Apparently, this problem can not be solved without exploring into the page content itself. Perkowitz et al. [57] proposed an approach, called conceptual clustering mining, which applies concept learning algorithms to refine and extend the usage-based clusters so that the discovered clusters are both cohesive in usage patterns and coherent conceptually. Here, conceptually coherent means the members of the same cluster are very similar in terms of the concepts (or topics) of their content.

Suppose the entire page set is D, and the discovered usage-based page clusters (e.g., using the PageGather algorithm) is denoted as a set C. Then, for each cluster c in C, a certain concept learning algorithm Γ (e.g., the GREEDY-3 algorithm [52]) is applied to find the concept of this cluster $v = \Gamma(c, D - c, L)$, where c is used as the positive example set, $(D - c)$ is used as the negative example set, and L is a pre-defined conceptual language (e.g., conjunctions of descriptive features) used to describe each page in D. Finally, each usage-based cluster c is replaced with its corresponding conceptual cluster c_v, which is the extension of v in D. Similar to usage-based clusters, these conceptual clusters can then be used to create synthetic index pages to assist users in their navigation.

There is also a limitation with the conceptual cluster mining approach: pages often co-occurring together does not necessarily mean that they are conceptually coherent. Therefore, conceptual cluster mining may exclude some useful patterns. However, conceptual clusters are much easier to understand. This is the tradeoff we have to make.

Oyanagi et al. [51] presented another approach, matrix clustering, which can also be applied to cluster web pages based on user access patterns. Matrix clustering is to extract dense sub-matrices from a large, sparse, binary matrix. Similar to association rule mining, matrix clustering also has the measures of support and confidence: support is defined as the area (or size) of the extracted sub-matrix, and confidence is defined as the density of the extracted sub-matrix. To apply matrix clustering to web log analysis, we need to create a binary matrix based on the relationship between users and web pages: each row represents a session, each column represents a web page, and each element has a value of 1 if the web page is in the session or 0 if not.

Clustering of User Transactions

Clusters of user transactions represent groups of user transactions that are similar to each other based on their access patterns. A simple method for measuring the

similarity between user transactions is to compare the pages accessed in them [47]. Suppose the entire page set is

$$U = \{url_1, url_2, \cdots, url_n\}$$

and each user transaction t is represented as a bit vector

$$t = \{u_1^t, u_2^t, \cdots, u_n^t\}$$

where each u_i^t is determined by the occurrence of url_i in t

$$u_i^t = \begin{cases} 1, & \text{if } url_i \in t \\ 0, & \text{otherwise} \end{cases}$$

then the similarity between two user transactions t and s can be simply computed as:

$$sim(t, s) = \frac{|t \cap s|}{\sqrt{|t| \times |s|}}$$

In addition to binary weights, featured weights can be used for u_i^t as well, e.g., weights based on page view-time and page access frequencies. Moreover, sequential information can also be combined into the clustering process if necessary.

Clusters of user transactions can lead to another kind of page clusters [47]. Given a user transaction cluster c, we can generate a representative virtual user transaction for this cluster

$$t_c = \{m_1^c, m_2^c, \cdots, m_n^c\}$$

where m_i^c is the mean of u_i^t from all user transactions in c

$$m_i^c = \frac{1}{|c|} \sum_{t \in c} u_i^t$$

After filtering out low-support pages (e.g., if $m_i^c < \mu$, then $m_i^c = 0$), t_c becomes a special kind of page cluster which represents the typical access pattern of a group of similar user transactions.

The abovementioned approaches represent each user transaction as a sequence of individual pages along with their features. Shahabi et al. [68, 67] proposed a more generalized model, called Feature Matrices (FM) model, which captures path segments instead of individual pages. An order-n path segment is defined as a length-n subsequence of pages in a path. Therefore, an individual page can be considered as an order-1 path segment. The FM model is a generalization of the vector model [64], and can capture different features of path segments, e.g., hit-count, relative location in the sessions, and view-time. A FM model of a given session is defined as:

$$U^{fm} = \left\{ M_{r^{n_1}}^{F_1}, M_{r^{n_2}}^{F_2}, \cdots, M_{r^{n_m}}^{F_m} \right\}$$

where each M is a feature matrix, each F_i represents a feature, r is the cardinality (number of pages) of the specific web space, and each n_i is the order of segments. Let $n = \max(n_1, n_2, \cdots, n_m)$, then U^{fm} is an order-n model. The order of a FM model determines the complexity of the model, and brings the capability to trade accuracy for performance, or vice versa.

Given two user transactions t, s and their FM models U_t^{fm} and U_s^{fm}, the dissimilarity between these two user transactions can be computed as:

$$D^F = \sum_{i=1}^{m} w_i \times D^{F_i} \quad \left(\sum_{i=1}^{m} w_i = 1\right)$$

where D^F is the dissimilarity between the two FM models U_t^{fm} and U_s^{fm}, and D^{F_i} is the dissimilarity between the F_i features of U_t^{fm} and U_s^{fm}, which can be simply computed using various similarity measures such as Projected Pure Euclidean Distance (PPED) [68, 67].

Clustering of Users

Clusters of users represent groups of users that may have similar interests while surfing the Web. There are at least two kinds of data that can be used for clustering web users. One is the user browsing pattern, including the pages being accessed, the way in which the pages have been accessed, and the queries used for searching, etc. Another one is the user personal profile, e.g., the user's sex, age, occupation, preferences, etc.

The user browsing patterns let us cluster users based on their actual activities, while the user personal profiles allow us to cluster users based on their background information with the assumption that users with similar background are more likely to have similar interests. However, as noted earlier, the users' personal information can only be obtained intrusively, therefore is rarely available.

2.2.2.4 Classification

The idea of classification is to map data items into a set of pre-defined classes. As previously discussed, content-based page classification is an important application of web content mining. In web usage mining the primary application of classification is to create user profiles based on their personal demographic information or access patterns [20]. The classification of users is similar to the clustering of users, only that the classification of users is performed with a well-defined class set, and probably a well-classified sample set as well.

It is also a possibility that classification can be applied to identify content pages purely from usage data. For example, one can identify content pages (also called *information content* pages, or IC-pages) based on URL properties (such as file size, file type, domain type, etc.) and a number of browsing features (such as page view-time, the unit view-time which is the page view-time divided by the file size, position of the page request in the user transaction, number of pages visited so far, etc.) [82, 84]. There also exists another approach, which first learns to identify *information content* words (IC-words) from the "browsing features" of the words (such as number of times the word is in a query text, number of times the word is in the anchor text of a followed hyperlink, etc.), and then discovers content pages based on these IC-words (e.g., by sending a set of selected IC-words to the search engines) [84, 83]. However, these two approaches both require a training data set in which every content page has been identified appropriately. This training data set can be obtained by conducting a specialized experiment in which, except for the logging of other necessary information, all the content pages are identified by the users manually.

2.2.2.5 Markov Models

Markov models [61, 24] can be used to predict the user's coming requests based on those requests made most recently in the same visit. In the domain of web usage mining, states in Markov models can be defined variously based on the pages already visited. Zukerman et al. [85] experimented with this idea by presenting a number of Markov models derived from aggregate user behaviors. For example, based on the first-order Markov model, the user's next request can be determined merely by the current position: $<D_{i-1} \rightarrow D_i>$, where D_i denotes the request at time i. A variation of this model assumes that the next request depends only on the referring document: $<D_{Ref_i} \rightarrow D_i>$, where D_{Ref_i} denotes the referer of D_i. Moreover, some second-order Markov models can be applied as well. For example, the next request can be determined by the last two requests: $<D_{i-2}, D_{i-1} \rightarrow D_i>$, or by the last request and its referer: $<D_{Ref_{i-1}}, D_{i-1} \rightarrow D_i>$.

Note that D_{Ref_i} is not necessarily the same as D_{i-1} due to the logging scheme currently used by most web servers. For example, if the user accessed a page sequence $\{A, B, A, C\}$, and the second A was accessed through the "Back" command in the client browser, then it could be missed from the server access log. Such being the case, given $D_i = C$, we have $D_{i-1} = B$ and $D_{Ref_i} = A$.

2.2.2.6 Specialized Learning Methods

Except for the aforementioned commonly used learning techniques, there also exist other learning techniques specialized for web usage mining. For example, Schechter et al. [65] proposed a special kind of abstraction of user behavior called path profiles, and presented an algorithm for creating these profiles from server access logs, as well as a number of methods to predict user requests based on these profiles.

A path profile is defined as a set of pairs $<p, c>$, where c is the frequency with which path p has occurred. Path profiles can be created by storing all paths in the form of a tree such that a walk down the tree is like a walk through a path of URLs. Except for the common root node, each node in the tree represents a page associated with an OccurrenceCount value, which is the number of times the path from root to this node has occurred. Moreover, the number of potential paths can be reduced at the time of storage by only considering paths with length greater than one and whose maximal prefix has occurred at least T times. Path Q is a maximal prefix of path P iff Q sequentially matches the first $(|P| - 1)$ elements of P. T is an empirical threshold that needs to be set experimentally.

The prediction of user requests based on path profiles can be performed as follows: suppose the current user session is $\{p_1, p_2, \cdots, p_n\}$, $\{p_i, \cdots, p_n, r\}$ is the most frequently occurring path with maximal prefix $\{p_i, \cdots, p_n\}$, and there exists no path with maximal prefix $\{p_{i-1}, p_i, \cdots, p_n\}$, then r will be predicted as the next request.

2.2.3 Applications of Web Usage Mining

After the learning process, the next step of web usage mining is to apply the user access patterns we have discovered to various web applications in order to improve their performance. There are at least four broad categories of such possible applications, including navigation optimization, search optimization, caching and throughput optimization, and the development of user demographic models.

2.2.3.1 Navigation Optimization

Navigation optimization means improving the ease with which web users can reach the contents of interest more quickly. Except for users' own bookmark collections, we can achieve this goal with three major approaches: history mechanisms, index and recommendation pages, and search.

History mechanisms are used to help users return to previously visited pages quickly, instead of navigating to those pages from scratch. Tauscher and Greenberg [75, 74] presented some experimental results on users' revisitation patterns, and proposed a number of history mechanisms as well as evaluations. However, history mechanisms are different from web mining approaches because no mining process is associated with them, even though some analysis is required to assess their feasibility.

Index pages are very popular in the Web environment, serving as various collections of URLs belonging to the same topic or a set of related topics. There are at least three kinds of such index pages:

- *Design-based* index pages hold links that are related from the view of the web designer.
- *Concept-based* index pages hold links that are potentially related because their contents are conceptually coherent.
- *Usage-based* index pages hold links that are potentially related because users often access them in the same visit (or sequential visits during a certain period of time).

A web site can be organized in various ways, based on different views of its contents. Design-based index pages are based on the designer's view, therefore are important and more appropriate for the organization and maintenance of the web site. Consequently, design-based index pages are typically presented as a web site's default interface, and are relatively static, i.e., should not be changed very often according to page contents or user access patterns. Concept-based and usage-based index pages are typically used only as complements of design-based ones. This means that concept-based and usage-based index pages are optional. Users may use these index pages only if they can not navigate efficiently with default design-based ones. Mixing all these index pages effectively is a very hard work, and will make the web interface much more complicated. However, it is believed that simple hierarchies and structures are always preferred for web design [41].

Index pages can be created and used in various ways. For example, design-based index pages are created manually by web designers and often used as static navigation pages, while concept-based and usage-based index pages are created automatically by certain learning programs and often used as dynamic navigation pages, e.g., through recommendations [47].

Usage-based index pages can be obtained from web usage data using various learning methods. For example, association rule mining and clustering can both be used for this purpose. A typical use of the usage-based index pages is through dynamic recommendations. For instance, with association rules we can make recommendations to users based on their previous requests in the same session. Since the association rules being used may change along with users' new requests, this kind of index pages have to be dynamic.

To obtain concept-based index pages, we have to look into the page content itself. For example, the previously discussed content-based page categorization and page clustering methods can be used for this purpose. The conceptual cluster mining approach [57] explores a different idea, which applies concept learning methods to further refine usage-based clusters in order to get clusters that are both cohesive in usage and conceptually coherent.

2.2.3.2 Search Optimization

Though we can discover navigation patterns from users' previous visits, and then use these patterns to provide guidance for new users, there is a significant problem here: merely because users took these access paths doesn't mean that they wanted these paths. They might not find what they wanted there. And it is extremely difficult to find out users' real interests without their cooperation, e.g., a content page survey. Therefore a more effective navigation method should be one with which users can tell the server what they want. Using search engines is currently the only way with which users can interact with the server and retrieve potentially relevant documents directly.

It is worth noting that search optimization is a different problem from navigation optimization. While the primary goal of navigation optimization is to help users reach the contents of interest *more quickly*, search optimization is focused on improving the quality of searching results, i.e., helping users find what they want.

In the previous section we have discussed various approaches for content-based search. In addition, we can do more by taking users' feedbacks into account: when users click some hyperlinks from the searching results and spend some time there, those hyperlinks will be given higher ranks for the specific searching keywords [26], and therefore can be pushed towards the top of the result list when those particular keywords are used for later query.

2.2.3.3 Caching and Throughput Optimization

The throughput of web accessing can be optimized with caching schemes or data pre-sending and pre-generating schemes.

Caching is the most widely used approach for throughput optimization. It can be categorized into two broad types: client caching and server caching. A typical client caching scheme is based on timing: each document sent to the client has an expiry time. When the document is requested again before the expiry time, the client will know that the document is unchanged and therefore should be retrieved from the client's local cache instead of being transferred again from the server. Therefore, except for saving the client's waiting time, client caching also saves the throughput at the server side. Client caching is applicable in web browsers as well as proxy servers.

Server caching focuses on a different purpose, which is to reduce the server's responding time by retrieving the requested document more quickly. For example, the server could maintain a cache that contains the most frequently or most recently visited documents. So whenever a cached document is requested, the server would be able to retrieve the document directly from the cache, which could be much faster than obtaining it from the disk or from other servers (e.g., database servers).

However, server caching gives rise to the data consistency problem, which is by itself a difficult problem and not discussed in this research.

Data pre-sending and pre-generating schemes assume that the server can predict users' requests, and then pre-generate dynamic contents or pre-send documents to the user before the actual requests are made. The key issue here is to make accurate predictions on the user's incoming requests based on those requests already made in the visit. As previously discussed, there exist a number of approaches for this prediction: association rules, sequential patterns and sequential rules [20], Markov models [85], and path profiles [65]. Because the prediction is not guaranteed to be correct, the server could generate or send unwanted documents for the user from time to time. Therefore, data pre-sending and pre-generating schemes actually save users' waiting time at the expense of higher workload at the server side.

2.2.3.4 User Demographics

User demographics can help us understand the web surfing preferences of a single user or a group of users. Although intrusive, user surveys are a very important data source for user demographics [60]. However, there is a limitation with this approach: the user specifications are probably satisfactory when gathered, but this satisfaction may degrade gradually because users' interests constantly change over time. Therefore, users should be able to, and have to, change their specifications from time to time.

Without surveys, user demographics can only be performed with certain assumptions. For example, we can assume that users from the same domain may have similar interests, and acquire the common characteristics, such as navigation patterns or frequently visited pages, of all the users coming from a particular domain. However, due to the high diversity of users' interests, this kind of common characteristics will be very difficult to acquire.

We can also partition users into different clusters or pre-defined classes based on their navigation patterns, the contents of the pages they visited, and the queries they used for searching. Each of such user clusters or classes represents a group of users that may have similar interests.

2.2.3.5 Summary

In the previous sections we have discussed four broad categories of web usage mining applications. However, the real-world applications are usually various combinations of them. The user customization (or user personalization) problem is an example of such applications.

Web customization is defined as any action that makes the web experience of a user customized to that user's taste [47]. First, we can provide customization based on users' own specifications, e.g., through surveys. If survey data is not available, then we can apply various learning methods to the usage data as well as the web content itself to extract knowledge useful for user customization. For example, we can create favorite indices for a specific user or a group of users based on their historical access patterns so that they can reach their most favorite pages more quickly. Also, we can provide dynamic recommendations based on users' current session paths to help users find pages of interest more quickly [47].

As previously discussed, survey-based user customization can be challenged by the continuous change of user interests. On the contrary, learning-based user customization can be updated automatically, along with users' visits and web content updates over time. Moreover, to perform learning-based user customization for a specific user, we can extract knowledge from the usage data of this user only, or from the usage data of aggregate users. However, sometimes it could be hard to gather sufficient usage data for learning from a single user.

Chapter 3

WebFrame — a General Framework for Web Mining

3.1 A Caricature of a Web Mining Problem Instance

Given a specific web mining problem instance, the pursuit of any possible solution requires us to address three basic requirements:

(1) an improvement measure.
(2) an abstraction or set of abstractions on the specific web space.
(3) a mechanism that supports the user's use of (2).

An ideal example, which is also the primary application that we originally obtained these observations from, is the problem of improving web navigation. The general idea of the problem is to improve a user's navigation within a particular web site by creating abstractions of typical usage. Those abstractions are created and subsequently used to guide a user's navigation.

For the problem of improving web navigation, a typical instance of an improvement measure is the number of hyperlinks traversed, and a typical form of (2) is a hyperlink graph representing a subset of the hyperlinked structure corresponding to a web site. A typical instance of (3) is any method which allows a user to exploit the abstractions — which could be something as simple as a pop up recommendation list, based on the hyperlink structure of (2).

This idea of making personal recommendations to each individual user based on the knowledge learned from aggregate user behaviors is also referred to as "collaborative filtering" [62, 69], which means filtering the information space to meet personal interests of individual users based on aggregate user behaviors. However, the primary idea of collaborative filtering requires the users to manually rank each object in the space (in our case, to label each web page as "relevant" or "irrelevant"), while in a general web mining system these labels are, more practically, assigned automatically by heuristics or models learned from a set of pre-labeled training samples.

One subtlety within this simple scenario is that there are an arbitrary number of methods to create the abstraction items (2), and that the specification of items (1) and (3) are not independent of any method of constructing instances of (2). In fact, when an abstraction is viewed as the result of a learning method, it is easier to

anticipate *some* impact of how one measures what is learned and how the results of that learning are to be deployed.

Another vital point is that item (3) is the component that embraces the idea of visualization. After all, it is a user that requires visualization aids to both understand the abstractions created by the learning methods, and to exploit them in a measurable manner. This applies for all kinds of users, including both web designers and web browsers. Of course there are as many methods of visualization as there are methods of learning, so again, the performance goals and potential methods of measuring their achievement are important to consider in the design of user visualization. We review some abstract dimensions of web visualization later in this chapter.

Even within this crude caricature, there are important questions to be asked and answered about an arbitrary web mining problem instance:

- Who is the improvement designed to benefit (e.g., individual user, class of users, web designer, Internet Service Provider, etc.)?
- What kinds of information can be gathered and used to build abstractions (e.g., static web site hyperlink structure, page content and meta-content, web logs, etc.)?
- What methods are most appropriate for a user (or class of users) to exploit the abstractions (e.g., navigation hints, site visualization, batch page retrieval and ranking, etc.)?

These are exactly the kinds of questions that we expect will continue to arise as we experiment with both the specification and use of our web mining framework — WebFrame.

3.2 System Architecture — WebFrame

Here we provide a high level description of the software architectural components that support the crude scenario above. Mostly we require enough precision to be able to anticipate a prototype application, without raising too many immediate principled objections.

Like existing web mining architectural proposals (e.g., [21, 47, 71, 81]), the gross level component architecture will require a module to support data capture, a module to support the specification and deployment of a repertoire of learning methods, and, perhaps less common, an explicit module designed to support evaluation of any combination of the first two. This basic architecture is illustrated by the diagram of Figure 3.1. Here the "Web Data Abstraction" is a generic concept representing the results of applying a learning method to web data.

In our particular instance, we have already extended the simple three component architecture into a more elaborate framework — WebFrame, as depicted by the diagram of Figure 3.2. This elaboration results from both top-down elaboration of the three component caricature, and from a bottom-up development of our current prototype.

One simple way to understand this instance of our architecture is to consider a high-level description of the process control within it. With respect to the diagram of Figure 3.2, our current web mining procedure can be described as follows:

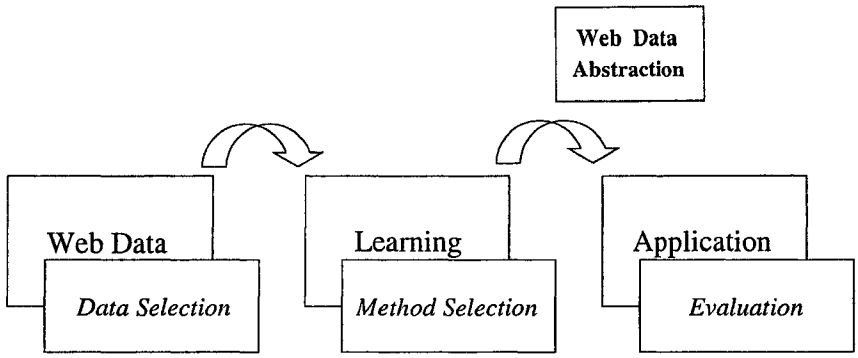


Figure 3.1: Basic Web Mining Architecture

1. Determine what data can be used, and obtain the data through the *Data Acquisition* module.
2. Transform the original data into specific formats that can be used by various learning methods. This work is done through the *Data Preparation* module.
3. Determine what learning algorithms are appropriate for the task, and then apply the algorithms to the formatted data to obtain certain knowledge. This work is done within the *Learning* module.
4. The knowledge obtained from various learning methods can be queried with certain constraints, like the query of a database. Moreover, the quality of the knowledge can be evaluated with detailed measures, e.g., by precision and recall. This work is done through the *Knowledge Analysis* module. The evaluation of the knowledge can then be used as feedback for making improvements in steps 1-3.
5. We can make use of the knowledge to achieve various tasks in certain applications, and evaluate the improvement of the performance. This work is done through the *Applications & Performance Evaluation* module. The evaluation of the performance improvement can again be used as feedback for steps 1-3.
6. Original data, formatted data, as well as the obtained knowledge can all be visualized through our *Visualization* module, to provide feedback for steps 1-3.

The most significant components of this web mining framework, which are also the two missing ingredients from most existing web mining architectures, include the integration of two modules: the evaluation of the performance improvement and the visualization module. These two modules are described in detail in the following sections.

3.3 Evaluation of Performance Improvement

Of note is the observation that no application of any learning method to the web data makes sense without first formulating a goal framework against which that method can be evaluated. This simple idea is typically the missing ingredient of many WWW mining techniques. Like many data mining methods, the weakest ingredient is the formulation of discovery goals. These goals are vital to the development of evaluation criteria for any learning method.

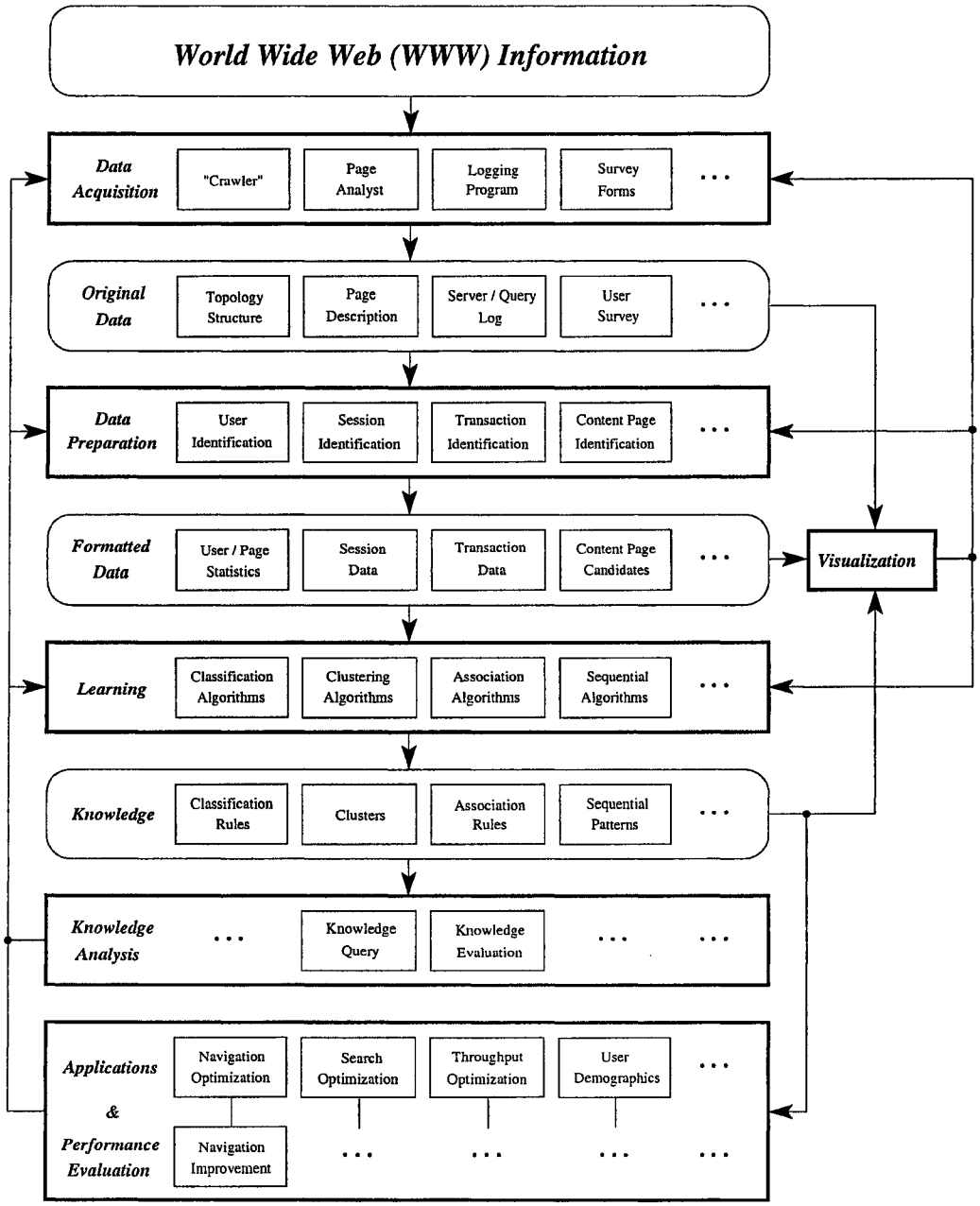


Figure 3.2: System Architecture — WebFrame

In our own case, our framework requires some evaluation methods that can provide a basis for determining the value of the web mining we undertake. In other words, we want to measure the improvement of various web applications after the web mining methods have been applied. We expect these measures to be application-specific, at least within the broad scope of navigation, search, caching and throughput, and user modeling. Most existing measurements for this purpose are qualitative, e.g., by the manual judgments of users or certain experts. The research on quantitative measurements for this kind of evaluation is still an open area.

In our experiments, we focus on the idea of web usage mining and the application of simple learning methods to improve user navigation. Even in this specific problem, there could be different kinds of evaluation measures for different kinds of user navigation goals. For example,

- If the user knows what and where to look at, then the appropriate evaluation should be: if we can help the user reach the target documents more quickly.
- If the user knows what to look at but does not know how to locate the documents, then the appropriate evaluation should be: if we can help the user reach the relevant documents more quickly.
- If the user is browsing the web site without a specific goal, then the appropriate evaluation should be: if we can help the user better understand the information contained in that web site.

In our case of using dynamic recommendations to improve user navigation in fixed document environments, precision and recall can be used to measure the quality of the recommendations. However, these two measures are not capable of evaluating the actual improvement in user navigation provided with help of the recommendation mechanism. In other words, an accurate recommendation is not necessarily a "useful" recommendation. For example, if the recommended page is exactly the page the user originally planned to visit next, the recommendation is actually useless.

The measure we used for evaluating the improvement of user navigation is called *navigation improvement*, which indicates whether we are actually "improving" the user's navigation by reducing the number of hyperlinks traversed to find "interesting" pages. Apparently, this evaluation is only applicable to users who have something "interesting" as goals while browsing the web site. Therefore, it is only applicable to the first two cases in the above example.

3.4 Visualization

Our framework for web mining includes no explicit commitment to any particular data format, but it is clear that the overall task will involve large volumes of data. For example, the total usage of the website of our academic department UACS (Department of Computing Science, University of Alberta) is 110~180GB per month, while the file size of typical web logs is about 600~700MB per month. Besides, the overall size of the UACS website is approximately 10~20GB (with 100,000+ HTML pages). It is relatively clear that any serious evaluation on such large data volumes will require some method of viewing these large volumes of data at some level of abstraction appropriate for humans, e.g., some form of visualization.

When important features and patterns of large data volumes can be distinguishing, the human problem solver can be more effective at guiding a problem solving tool inside of a large search space. And within our framework, we require some kind of visualization in order that the human user can help understand and evaluate the results of any particular learning method. An example of using visualization to aid in knowledge discovery is shown in the WebLogMiner system [81], which has some built-in visualization functions based on data cube and OLAP techniques. However, in WebFrame we intend to build an independent and more sophisticated visualization tool which can be used to visualize both web site structure and web usage.

Within the scope of our current framework, we intend to provide a human user with help in at least two areas:

- a visualization that helps a user maintain a sense of context within the web navigation space, and
- a visualization that provides an external representation of both static and dynamic navigation, to permit visual comparisons of different accumulated web navigation trails.

Although these two kinds of visualizations can be presented separately, they are also inevitably related to each other. The ability to show how a web site is organized *as well as* how the web site was used by the users is important to understand the correlation between web usage and web site design. Therefore, we have developed our visualization tool to provide insight into both web site structure *and* web usage. While our tool is constantly evolving under our own use and experiments, our current version provides the ability to view individual web sites at different levels of granularity, and allows both the static and dynamic display of individual and aggregate user behavior.

Our own web visualization tool, WebKIV, has been designed to provide a uniform method of visualizing web data along two dimensions. We are attempting to develop a single visual foundation within which one can visualize both static and dynamic structure, as well as both the individual and aggregate behavior of web users. Our desire is to try and combine the features of both such systems. We understand the compromises, but hope that the advantages of being able to concurrently visualize web sites *and* web site usage will compensate for any loss of elegance apparent in visualizations that separate these functions.

In what follows we briefly discuss the visualization of both web site structure and web usage data using our WebKIV tool. For more information about this web visualization tool, please refer to [50, 49].

3.4.1 Web Site Structure Visualization

To facilitate pattern finding and extraction, our WebKIV tool uses the linear magnification to drill down, observing the data in detail. Our approach to web site structure visualization is relatively straightforward, and is loosely based on various two-dimensional displace techniques that focus visual attention in two spaces on a single URL, from which links are radially drawn outward. In particular, we use a version of Disk Tree, the radial tree algorithm of [16], to provide a two-dimensional display of an arbitrary URL. Since our goal is a general visualization of both structure

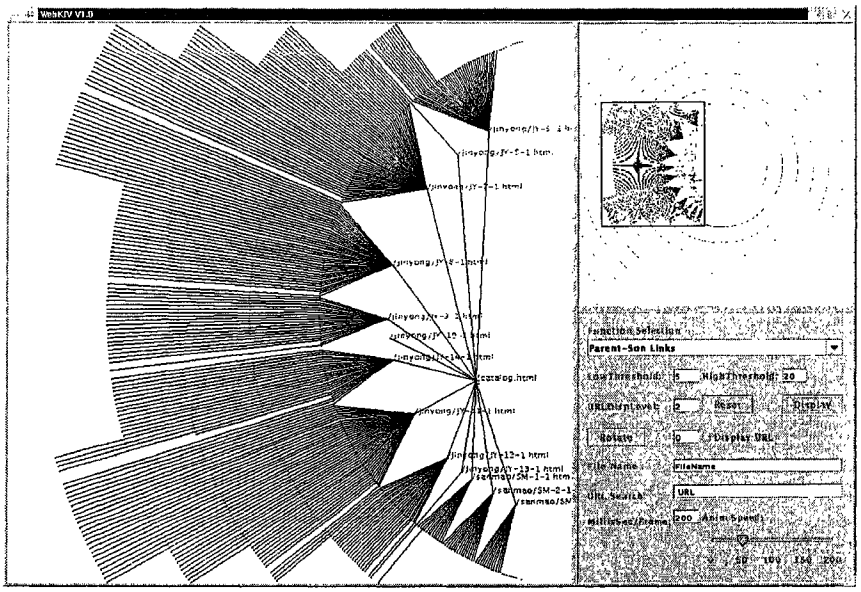


Figure 3.3: A Screen Shot of WebKIV — Our Web Visualization Tool

and usage, we take this decision to help reduce the complexity of our experimental visualizations.

A screen shot of our WebKIV tool is given in Figure 3.3. The radial tree is a hierarchical acyclic tree, with each level of the hierarchy represented by a concentric ring of page nodes, distributed around a central focus node. The focus node is designated by the user as a starting URL, which provides the initial focus for any visualization display.

Similar to heights on a topographic map, nodes linked from the focus URL are displayed on different concentric circles according to their levels in the hierarchical tree. A breadth-first search that avoids cycles is used to traverse the web site, to build the hierarchical structure. The radial pattern of links from a given node is represented by a radial distribution of nodes on a concentric circle, proportional to the number of nodes. Note, for example the concentric rings of linked nodes in the diagram of Figure 3.3.

This two-dimensional display based on concentric link "isobars" does not provide any insight to web structure, without an ability to shift the focus and level of detail. With this in mind, our visualization tool also provides the user with the ability to drag a rectangle over the visualization, then zoom on that rectangle. As shown in Figure 3.3, a smaller window (context window) always provides a context for the web site structure, while the other larger window (focus window) displays the detail of choice, which is a subset of the larger context. Whenever there is a need to drill down, the selected sub area is enlarged, and both windows are appropriately modified. If the user "zooms in" to sufficient detail then individual URLs are used to label each node.

3.4.2 Web Data Visualization

The visualization of web structure has many potential applications on its own. For example, the radial distribution of nodes at the same level of detail provides some

notion of page distribution over links. Web designers can use this kind of structural information to adjust the distribution of pages, e.g., with page hit rates to redistribute pages.

But if more detailed usage is important, then it is important to be able to visualize user navigation paths, either statically or dynamically, in order to be able to appreciate *how* a user traverses a web site.

If we can visualize *individual* user navigation paths superimposed on top of the web site visualization described above, we can begin to recognize well-traversed paths. These might be links that are popular over some particular time period, or trajectories of heavily visited web pages which help us understand how users arrived at web site "hot spots." In addition, we can visualize the distinction between "hub" and "authority" page types, from the user's navigation patterns of entering, viewing, and leaving web pages.

In addition to visualizing the navigation paths of individual users, we can also visualize aggregate paths. For example, when an individual traversal of a hyperlink is indicated by drawing a line from one node to another, the aggregate behavior of two users can simply annotate that link for each traversal. There are an arbitrary number of ways to visualize aggregate link traversal, e.g., by line width, color, annotation, etc. But when we can visualize *aggregate* user navigation paths superimposed on top of the web site visualization, we can begin to recognize web navigation clusters in order to understand aggregate user behavior. For example, this is useful to validate web site design, by statically viewing the distribution of aggregate navigation paths on a web site. And with appropriate navigation path annotation, we can dynamically observe aggregate behavior, e.g., aggregate navigation path changes over different time periods.

Moreover, within the context of our web mining framework, the goal of improving navigation requires us to visually compare different methods of navigation, which we describe later in the experiment section.

Chapter 4

Experiments with Navigation Optimization — a Web Usage Mining Instance

4.1 Navigation Optimization — a Web Usage Mining Instance

Within the growing literature on web mining, there is a relatively coherent thread of ideas focused on improvements to web navigation [21, 47, 56, 65, 71, 85]. Our first experiments with our web mining framework have the same goal of improving user navigation on the Web.

Navigation Optimization (also called Navigation Compression) means improving the ease with which users can reach the contents of interest more quickly. The basic idea is that we can discover navigation patterns from previous visitations, and then use these patterns to provide guidance for new visits. As described later, we call our learned "patterns" Navigation Compression Models or "NCMs."

Our web mining framework can help us gather various web data, learn from the data, and adjust the navigation structure of a web site to provide more direct access to appropriate pages. As discussed previously, this adjustment is performed by adding automatically created concept-based or usage-based index pages, instead of changing the original design-based structure of the web site. In other words, we attempt to help users in their navigation without changing the actual contents and the web structure.

4.2 Navigation Compression Models

In the deployment of our web mining framework to the navigation optimization problem, the missing middle component is that object to be created by various learning algorithms, and then inspected to see whether the learning algorithm has found something "interesting" which can provide navigation improvements, as measured by appropriate evaluation methods. We call the objects created by the application of learning methods *Navigation Compression Models* (NCMs).

The idea and name arise from the principle behind learning. All learning creates

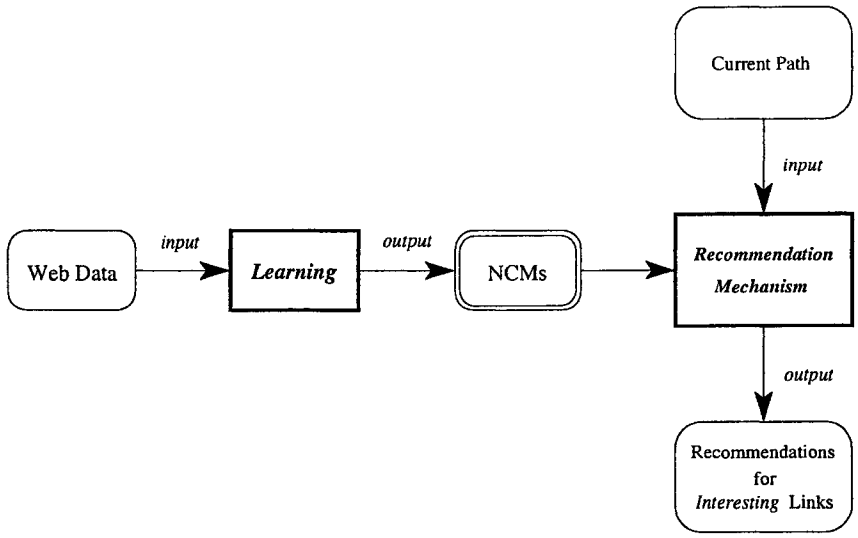


Figure 4.1: Using NCMs for Dynamic Recommendations

an abstraction of the initial input data, which somehow represents the input data in some other representation. For example, a classification hierarchy over a number of input instances, is a compressed representation of that input.

Similarly, our navigation compression models (NCMs) are simply some representation of an abstract navigation space, determined as a function of selected attributes of a collection of user navigation paths on actual websites. From this point of view, our NCMs are determined not only by the content and structure of the web site, but also by the potential interests and navigation behaviors of the users.

It is worth noting that the idea of NCMs, though originally arising from learning, is in fact broader than that. For example, some web site may provide a number of pre-designed navigation templates to aid in different kinds of user interests. These are another kind of NCMs which represent an abstract navigation space determined from envisioned instead of real user navigation paths. Such NCMs are not created automatically by learning but designed manually.

4.2.1 NCMs for Dynamic Recommendation

We are interested in experimenting with various learning methods, and comparing their performance on navigation improvement by comparing navigation compression models. We can evaluate our NCMs statically by measuring overall navigation improvement in terms of reducing the number of hyperlinks traversed by an individual user to find "interesting" pages.

To actually employ the NCMs, we can use one of two approaches: one is using synthetic static index pages, each of which contains indices to a set of pages belonging to similar or related topics; another one is using dynamic recommendations, which use NCMs to make recommendations to the user based on the pages the user has already visited. In our experiments we use NCMs to implement dynamic recommendations, as depicted in Figure 4.1.

Each NCM used for dynamic recommendations can be formulated as a function

Table 4.1: Example of Different NCMs and Associated Prediction Schemes

association-rule-based NCMs vs. cluster-based NCMs

Type of Knowledge	Prediction Scheme
association rules $r = \{a_1, \cdots, a_m \rightarrow b_1, \cdots, b_n\}$ a_i, b_j : web pages	If pages in the antecedent of the rule all appear in the current user path, then those unvisited pages in the consequent of the rule are predicted as the user's interested documents.
clusters $c = \{p_1, \cdots p_l\}$ p_k : web pages	If one or more pages in the cluster appear in the current user path, then those unvisited page in the cluster are predicted as the user's interested documents.

that generates recommendations based on a given path:

$$NCM = f(path \Rightarrow recommendations) \tag{4.1}$$

where the recommendations can be as simple as the frequently visited pages, or as complex as the predictions of web documents that might satisfy the user's incoming requests.

Note that there is a significant problem with user navigation patterns: a traveled path is not necessarily a desired path. Therefore we propose a recommendation mechanism which ignores those auxiliary pages and makes recommendations only on "relevant" pages. Of course the identification of "relevant" pages is a very difficult problem, but we can make assumptions based on certain heuristics. In addition, although the identification of the relevant pages is heuristic, we can still compare NCMs for how well they can potentially help users find relevant pages more quickly.

4.2.2 NCMs in an Unique Form?

As mentioned earlier, each NCM can be described as a function with a user's navigation path as input and predictions of the user's interested documents as output. Based on this definition, a NCM requires at least two basic components: (1) a knowledge base containing navigation patterns learned from historical user navigation paths, and (2) a prediction scheme specifying the mechanism of the prediction process using the knowledge base.

There are a number of learning techniques that can be used for creating NCMs. Different learning techniques generally produce different forms of knowledge, which accordingly requires different schemes to perform the prediction. For example, association rules and clusters are in different forms, therefore are associated with different prediction schemes, as shown in Table 4.1.

A question then arises regarding the form of NCMs: "Can we generalize NCMs created from different learning techniques into an unique form?" If such a generalization can be accomplished, then we can use a universal prediction scheme for all the generalized NCMs. Moreover, when there comes the need to adapt the prediction scheme to various web sites or user preferences, we can make changes only to the

universal prediction scheme instead of making changes to each prediction scheme for different kinds of NCMs.

The generalization of different kinds of NCMs is still an open problem. In an extreme case, we can define a generalized form for NCMs based on the description in Equation (4.1) that correlates each possible navigation path with a set of predictions. This approach is apparently impractical because of the huge (if not infinite) number of potential navigation paths. Currently we have found no generalized form of NCMs that can be practically used. Therefore, in our experiments, we use different kinds of NCMs in their own forms and with their own prediction schemes.

4.3 Experiment Methodology

Our experiments focus on the idea of learning from web usage data to improve user navigation. While we have various combinations of data and learning methods to select from, a proper evaluation method is crucial to accurately assess the improvement of the navigation. However, there has been little effort directed at this problem, and so far we know of no evaluation method which can evaluate navigation improvement quantitatively and visually.

The basic idea behind such an evaluation is that we can compare the compressed navigation paths with the corresponding paths without compression. However, we can not expect to obtain both of these navigation paths from the same user without the previous navigation experience somehow affecting the later one. Such being the case, we can envisage an ideal experiment conducted as follows:

> Suppose we have a group of user subjects with the same or similar educational background, similar interests, and the same level of web experience. Each subject is asked to fulfill a fixed number of tasks. Each task can be described as finding some specific information starting from a given entry point. Moreover, each task is randomly assigned to a fixed number of users such that half the users are helped with the navigation compression mechanism, and half the users are not. In this way we can collect compressed navigation paths together with the corresponding uncompressed paths from different, but similar, users.

Such an experiment involves certain problems like user subject composition and task distribution, but these problems are not impossible to solve. However, our research objective is slightly different: we want to make maximal use of the data at hand without intrusive data collection, even for the purpose of evaluation. Based on this consideration, our experiments are designed to use the web log data for both training and testing, as shown in Figure 4.2. With respect to this idea and the web mining framework proposed in Section 3.2, the experimental procedure can be described as follows:

1. Transform original log data (both training and testing) into sessions. As previously defined, a session is a sequence of page accesses from a single user during a single visit to the web site. We acknowledge that this work involves enormous challenges in "heuristic" identification of users, sessions, content pages, etc.

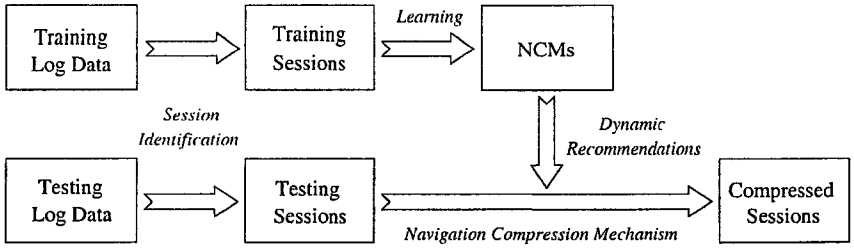


Figure 4.2: Experiment Methodology

2. Apply a selection of learning algorithms to the training sessions to obtain various kinds of NCMs. As mentioned earlier, these NCMs have the ability to predict the user's possible interests and therefore can be used for making dynamic recommendations. The learning algorithms that can be used for this purpose include association rule mining, sequential rule mining, clustering, Markov models, etc.
3. Apply the NCMs to the testing sessions through our navigation compression mechanism, and generate a new set of compressed sessions. There are two important issues that need to be addressed in this navigation compression mechanism: (1) the generation and use of recommendations, and (2) the simulation of user actions to the recommendations.
4. Finally, the value of the NCMs is determined by using a family of evaluation methods. A basic method evaluates navigation improvement merely based on the difference between the number of traversal links of the compressed sessions and that of the corresponding original sessions, while a more comprehensive method also takes cost of recommendations into account.

In this procedure, to actually perform navigation compression, we need a model that can be used to simulate the user behaviors in using dynamic recommendations. In our experiments, we simply assume that the user will follow all the "correct" recommendations. The meaning of "correct" will be described in detail later in the evaluation section. Due to this assumption, the measured navigation improvement is only an estimation of the potentially best improvement that can be achieved.

Our experiment is divided into three parts, corresponding to the three basic components of the web mining framework:

1. Data Selection and Preparation
2. Learning
3. Evaluation

In the following chapters we present in sequence our experiments on each of these parts, as well as the analysis of the results. The data set used in these experiments is obtained from the server access log of our academic department — UACS (Department of Computing Science, University of Alberta).

Chapter 5

Data Selection and Preparation

While the ultimate goal of our experiments is to assess the value of various learning methods in the application of improving user navigation, the proper selection and preparation of experimental data is as important as the learning methods themselves. In fact, the choice of learning methods will itself suggest constraints on data gathering and preparation.

There exist a large variety of user communities throughout the Web. While users in different communities may have different backgrounds, interests, preferences, and usage patterns, it is reasonable to expect that the same learning methods may have different impact on different user communities. Moreover, the web design style, web structure, as well as web content, may also have impact on user navigation behavior. Therefore, the experimental result from each data set is only applicable to the specific user communities and web space the data set represents, and other similar web environments.

Understanding and interpreting the original data is the key issue in data preparation. The following questions indicate the major concerns in this task:

- "What is the meaning of the data?"
- "What kind of knowledge can we expect from the data?"
- "Is the data sufficient to obtain the knowledge we expect, or is additional data required?" and "Can we do better with more data?"
- "Is the data appropriate to apply learning methods to, or is preprocessing required?" and "If preprocessing is required, is the data clearly self-explained, or assumptions have to be made?"

5.1 Data Selection

In the experiments, we have collected 24 months of access logs from the UACS (Department of Computing Science, University of Alberta) web server, from January 1, 2001 to December 31, 2002. The major user communities using this web server include UACS faculty, staff, graduate students, undergraduate students, and external users.

The access log format used by the UACS web server is *Apache HTTP Server's combined log format + "cookies"*. The Apache HTTP Server's combined log format has been described in Table 2.1. The cookies are <name, value> pairs generated

```
ac8e8c53.ipt.aol.com
-
-
[07/Sep/2002:18:19:15 -0600]
"GET /~tongz/ta/391/cmput391.html HTTP/1.1"
200
2695
"http://www.cs.ualberta.ca/~tongz/"
"Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)"
"uabcs=ac9ce052.ipt.aol.com.181501029729097960; RT_id=688618486"
```

Figure 5.1: UACS Access Log Example — One Log Record

by the server for various purposes, e.g., user tracking. Each cookie is defined in the form of

$$cookie\_name = cookie\_value$$

and different cookie definitions are separated with ";". In the UACS access logs, a cookie named "uabcs" is used specifically for user tracking. An example of the UACS log record is given in Figure 5.1.

While the use of cookies can dramatically increase the accuracy in identifying anonymous web users, some users might consider it an intrusive method of data gathering. However, "intrusive" is hard to characterize and may have different levels. Though the collection of cookies might be more intrusive than pure anonymous web logging, it is much less intrusive than those data gathering methods that require user interaction, such as user authentication and user surveys. Therefore, there also exist some other users who do consider cookies non-intrusive. In our experiments we use cookies for user identification because the server access logs we use for web mining already contain this information.

Our data preparation process consists of breaking the access log into user sessions, and the identification of content pages. In what follows we detailedly discuss these two issues in sequence.

5.2 Session Identification

We have previously mentioned various approaches, as well as corresponding difficulties, for partitioning the access log data into user sessions. The key issues of this work include: (1) user identification, (2) session boundary determination, (3) page cleaning, (4) path completion, and (5) page view processing.

5.2.1 User Identification

As noted earlier, there are currently many ways to identify each user, from non-intrusive to intrusive ones. In our experiments, we designed a mechanism which uses the following data for user identification: user authentication, user-tracking cookie, IP address or domain name, user agent, referer, and site topology. With these data, user identification can be performed in the following steps, depending on the data we can gather:

1. Users can obtain authentication through registration. With this information, users can be identified 100% accurately. Unfortunately, this data is not widely available, and is highly intrusive.
2. If user authentication is not available, we can use a cookie specifically designed for user tracking. The accuracy of this method is also very high. However, this data is also not always available, because the users have the option to reject cookies in their browsers.
3. If both user authentication and user-tracking cookie are not available, the next choice is to use the IP address or domain name. The accuracy of this method is much lower then the previous two. Such being the case, the user agent information can be used together with the IP address to improve the accuracy. Different user agents indicate different users, even if they have the same IP address.
4. The referer can be used to further improve the accuracy of user identification. Suppose a user session already contains a path $P = \{r_1, r_2, \cdots, r_n\}$. Then a new request r from the same IP address belongs to the same user if $\exists r_i \in P$, such that $r_i = referer(r)$. Otherwise, r belongs to a different user.
5. If the referer information is not available, we can use the site topology instead. Suppose a user session has already contained a path $P = \{r_1, r_2, \cdots, r_n\}$. Then a new request r from the same IP address belongs to the same user if $\exists r_i \in P$, such that $M(r_i, r) = 1$ (there exists such a link $\{r_i \rightarrow r\}$), where M is the link matrix representing the site topology. Otherwise, r belongs to a different user.

Based on the UACS access log format, we have the following two user identification methods to choose from:

cookie-only This method identifies users by their authentication or user-tracking cookie, and excludes all log records without both pieces of information. This method is named after the idea of a cookie, because the user authentication is not available in most log data.

ip-link This method identifies users with all the information we have, and no log record is excluded.

Although the *cookie-only* method has higher accuracy, it will miss the log records from users rejecting cookies. Moreover, even if the user accepts cookies, it is still possible that the first record of some session has no cookie value, e.g., when the user accesses the server for the first time, or when the previously assigned cookie value has expired. In this case, the first requested page can still be discovered from the "referer" field of the next request, but its viewing time information will be lost.

The *ip-link* method is less accurate, but generates more data for learning. In case of the first record problem mentioned above, this method may identify two users instead of one: one with cookie, and one without. And the latter one will have only one record — the first record, in the session. Such being the case, we need to combine the two users into one user, e.g., by comparing the IP address and user agent information.

5.2.2 Session Boundary Determination

As discussed earlier, we can use a timeout to determine the session boundary. We call this method *timeout-based* session identification. Using this method, we can determine the session boundary as follows:

> Suppose a user session has already contained a path $P = \{r_1, r_2, \cdots, r_n\}$. Then a new request r from the same user starts a new session if $t(r, r_n) > \varepsilon$, where $t(r, r_n)$ is the duration between request r and r_n, and ε is a predefined timeout.

It is worth noting that the timeout primarily refers to the viewing time only. But here, $t(r, r_n)$ is actually the sum of the transfer time and the viewing time. However, we ignore the transfer time to simplify the problem. In our experiments, we arbitrarily select a widely-used session timeout setting: $\varepsilon = 30$ minutes.

However, based on the UACS log format, we do have another choice which does not need to explicitly specify a timeout to determine the session boundary. We call this method *link-based* session identification. Using this method, the session boundary is determined merely based on the "referer" information in the log:

> For a specific user, if a page request has no "referer", it is assumed that the user is starting a new session.

Both of these two methods have their limitations. For example, the timeout-based method will miss the case when the user leaves for a while and comes back later continuing the left work. And the link-based method may incorrectly combine multiple sessions into one if the user always leaves a default page open, and starts every surfing from that default page.

5.2.3 Page Cleaning

The task of page cleaning is to select appropriate types of documents for analysis and exclude documents of other unwanted types. A common way of doing this is to discard all the image files (e.g., GIF and JPEG files) because they are generally associated within some main documents (e.g., HTML files). Of course which types of documents are interesting varies according to the contents the website is providing. For those image-providing websites, image files are certainly interesting and should not be excluded for analysis.

In addition to image files, there are many other types of documents in the log as well, e.g., postscript and PDF files. To simplify the problem, in our preliminary experiments we keep only HTML pages. The documents of other types, even though potentially useful, are only one link away from their corresponding HTML pages.

HTML pages can be classified into two broad categories: static pages and dynamic pages. Static pages are pages with relatively static content which can only be updated by hand, while dynamic pages are pages containing dynamic content which can be generated on the fly while being requested. There are many technologies for producing dynamic pages, such as CGI scripts, Server-Side Includes (SSI), ASP scripts, PHP scripts, etc. It is hard to estimate the rate between static and dynamic pages. But it is widely believed that the number of dynamic pages is much larger than that of static pages, and is still growing at a fast rate.

When the rate between static and dynamic pages in a specific web site is hard to obtain, we can estimate the rate between user-accessed static and dynamic pages from web usage logs. This is done by counting only HTML pages (static or dynamic), and excluding those auxiliary documents (e.g., image files, postscript and PDF files, etc.). For example, from the UACS log data of October 2002, we know that during this period the users had accessed approximately 39820 static HTML pages and 18473 dynamic HTML pages. Moreover, these 18473 dynamic pages were actually generated from 1099 distinct script files with various parameters. This statistic was also calculated on the UACS log data of other periods, and the results show that in the UACS web site, requests for static pages make up a significant portion of the web usage logs. Therefore, dynamic pages are not necessarily accessed more often than static pages, even though dynamic pages have a potentially larger volume.

Although dynamic pages may also be useful, recommending on dynamic pages could be a very complicated task. The dynamic content may change quickly which could lead to an inconsistency between the page being recommended and the page analyzed before. Moreover, many dynamic pages are parameter-driven. Providing different parameters to the same script results in different dynamic pages. Therefore, by including such kind of dynamic pages there could be too many distinct pages that will make the pattern extraction much more difficult. Such being the case, for simplicity we concentrate only on the static pages in our experiments. This is done by only keeping those pages ending with ".htm", ".html", and some of their alternatives (e.g., pages ending with ".ehtml", ".jhtml", ".shtml", ".phtml", etc.) without parameters.

5.2.4 Path Completion

In our experiments, path completion is quite simple. Given a user path $P = \{r_1, r_2, \cdots, r_n\}$, a new request r from the same user, and the referer of the new request $referer(r)$, if $referer(r) \neq r_n$, then the user path after path completion should be $P' = \{r_1, r_2, \cdots, r_n, referer(r), r\}$.

5.2.5 Page View Processing

We have previously discussed the idea of page view, with which we hope to exclude log records of those associated documents, and only keep the record of the main page for pattern analysis.

Since we have kept only HTML pages after the page cleaning process, what is left for the page view problem are the multi-frame pages. In a multi-frame HTML page, each frame is also an HTML document associated with the main page, and would be requested automatically whenever the main page is requested.

As noted earlier, page views can be extracted with simple analysis of the content but page content is not always available. Therefore, in our experiments we propose a pattern matching approach to identify page views merely from the web logs. Generally a request of the main page of a page view can be captured with a generalized form of navigation sequence in the log:

$$A[t] \rightarrow A_0[t] \rightarrow \{A'[u] \rightarrow A_i[t]\}_n \rightarrow A'[u] \rightarrow A_{n+1}[T] \quad (i = 1, \cdots, n\,;\, n \geq 0)$$

In this generalized sequence, $A[t]$ is a request of page A with viewing time t, $A'[u]$ is a revisit of page A with unknown viewing time, and $\{A'[u] \rightarrow A_i[t]\}_n$ means

the pattern $\{A'[u] \rightarrow A_i[t]\}$ can be repeated multiple times. Moreover, t is a very short viewing time (e.g., no more than two seconds) with which we can assume that the next request of A was probably generated automatically by the browser because of the page view, and T can be any viewing time value. As an example, given a page view in which A is the main page and $\{A_1, A_2, A_3\}$ are the associated pages, a request of page A would generate a sequence of requests in the log as follows:

$$A[t] \rightarrow A_1[t] \rightarrow A'[u] \rightarrow A_2[t] \rightarrow A'[u] \rightarrow A_3[T]$$

Our experiments showed that the page views identified using this approach (with $t = 2\,seconds$) are almost 100% accurate. Setting t to a longer viewing time will result in a larger number of identified page views but decrease the identification accuracy. Obviously there is a tradeoff between the number of page views identified and the accuracy of the identification.

After identifying page views, how to represent them in the sessions is also a difficult problem. First, we want each session record to represent a true navigation action by the user. Therefore we can not leave the original sessions unchanged because in those sessions each page view is inappropriately represented as a navigation sequence. Moreover, in original sessions the viewing time of an entire page view is always assigned to the last requested associated page, and every other page only has a very small viewing time close to zero.

As previously discussed, we can combine all the log records of a page view into a single request of its main page. In this way the viewing time of the entire page view can be simply assigned to the main page. However, there is a problem with the follow-up requests. For example, given a page view with main page A and associated pages $\{A_1, A_2\}$, suppose an original session looks like follows:

$$A[t] \rightarrow A_1[t] \rightarrow A'[u] \rightarrow A_2[T] \rightarrow A'_1[u] \rightarrow B[T] \rightarrow A'_1[u] \rightarrow C[T]$$

After combining all the log records of the page view, we have:

$$A[T] \rightarrow A'_1[u] \rightarrow B[T] \rightarrow A'_1[u] \rightarrow C[T]$$

This is where the problem arises. In this combined session, $A'_1[u]$ is the result of the path completion process but may not be the actual request we want. Based on the definition of path completion, this sequence can be interpreted as follows: the user went to page A, then revisited page A_1, then went to page B from A_1, then revisited page A_1 again, and then went to page C from A_1. However in this example, since A_1 is an associated frame of A, the first revisit to page A_1 never really happened. Moreover, if page B was targeted in the frame of A_2, then the second revisit to page A_1 did not really happen neither.

To solve this problem, after combining the log records of page views, we further refine the combined sessions by simply removing any revisited page which is an associated page of a previously-visited page view. In the above example, the refined session should be:

$$A[T] \rightarrow B[T] \rightarrow C[T]$$

This solution is based on the assumption that any such revisit to an associated page did not really happen. However, this is not always true. In our example, whether the second revisit to page A_1 is true depends on where page B was targeted. Unfortunately this information is not recorded in the logs.

5.2.6 Experiments

Through some experiments we hope to compare the results of the various user and session identification methods mentioned above, and if possible, determine which choice is better. The experiments were conducted by generating sessions on each set of monthly UACS logs, from January 2001 to December 2002. Among all the sessions generated, we are only interested in those with length between 2 and 100. This is for two reasons: first, a session has to have at least two requests to be useful for pattern extraction; second, those extremely long sessions are considered as exceptional circumstances, and therefore should be excluded for pattern extraction. In our experiments, we arbitrarily specify 100 as the maximum length of "useful" sessions. Of course, this constraint can itself be an experimental parameter in a more comprehensive experiment.

5.2.6.1 Experiment 1: User Identification Methods

The first experiment was conducted to compare the two user identification methods: cookie-only method and ip-link method. For session identification we used the timeout-based method, with the timeout being 30 minutes. The experimental results are shown in Table 5.1.

The results show that with the ip-link method, we can identify a lot more users and generate a lot more sessions. But if we only keep those sessions with length between 2 and 100, then the difference between results from the two methods becomes quite small. In most cases, the difference in number of sessions (with length between 2 and 100) is less than 8%. This is illustrated in Figure 5.2. Moreover, the difference in average session length is also very small, which is less than 2% mostly.

Based on these results, in the following experiments of navigation improvement we choose to use only the cookie-only method for user identification, because this method can prepare sessions much more accurately without losing a significant proportion of the data.

In addition to the above results, we also found that most users choose to accept cookies, not rejecting them. In the experiment we found that in most monthly data sets, the proportion of log records with cookie values was larger than 80%. This also supports our statement that we can use the cookie-only method without losing a significant amount of data.

This experiment also showed that most sessions obtained from non-cookie records (80% of such sessions) contain only one request, therefore are not useful for navigation improvement. These log records might come from some web robots or crawling programs.

5.2.6.2 Experiment 2: Session Identification Methods

The second experiment was conducted to compare the two session identification methods: timeout-based method and link-based method. For the timeout-based method, the timeout is still set to 30 minutes. For user identification we used the cookie-only method. The experimental results are shown in Table 5.2.

The results show that with the link-based method, we can generate fewer but longer sessions. However, the difference between results of the two methods is quite small. If we only keep those sessions with length between 2 and 100, then the

Table 5.1: Experimental Results for User Identification Methods

cookie-only vs. *ip-link*

Term	Number of Users	Number of Sessions	Number of Sessions ($1 < length \leq 100$)
01/2001	55709 / 77857	146059 / 197519	79373 / 89845
02/2001	37231 / 54863	155300 / 187525	74970 / 79265
03/2001	40509 / 53784	173135 / 194433	83486 / 87416
04/2001	37409 / 52946	153813 / 177614	75687 / 81519
05/2001	35634 / 63109	120850 / 172636	61125 / 89189
06/2001	28400 / 39474	93123 / 108635	46225 / 50130
07/2001	30286 / 40276	95561 / 110672	49198 / 51320
08/2001	33538 / 43908	112402 / 128127	57011 / 59452
09/2001	38323 / 50153	208162 / 226668	93309 / 96362
10/2001	46428 / 60381	226526 / 249431	115384 / 119548
11/2001	47816 / 62942	200478 / 225482	103216 / 107747
12/2001	39392 / 53816	170163 / 191506	85046 / 90399
01/2002	46372 / 63194	215795 / 242432	106389 / 111720
02/2002	44920 / 60370	235623 / 259359	111424 / 115660
03/2002	49623 / 72678	212619 / 245241	104221 / 110176
04/2002	47997 / 64026	195312 / 219230	95746 / 100488
05/2002	44546 / 58890	145958 / 167543	72547 / 76363
06/2002	40143 / 51873	127465 / 145087	70389 / 74031
07/2002	40602 / 53162	129175 / 148498	69288 / 72535
08/2002	36641 / 49441	138355 / 157773	70443 / 73717
09/2002	47071 / 71905	249328 / 328140	118029 / 121944
10/2002	52836 / 73725	242430 / 277088	113755 / 120300
11/2002	59263 / 78978	222307 / 254520	100846 / 108116
12/2002	40750 / 53356	193534 / 213199	91200 / 94368

Note: Each value in this table is represented as $<V_c/V_i>$, where V_c is the result from the cookie-only method and V_i is the result from the ip-link method.

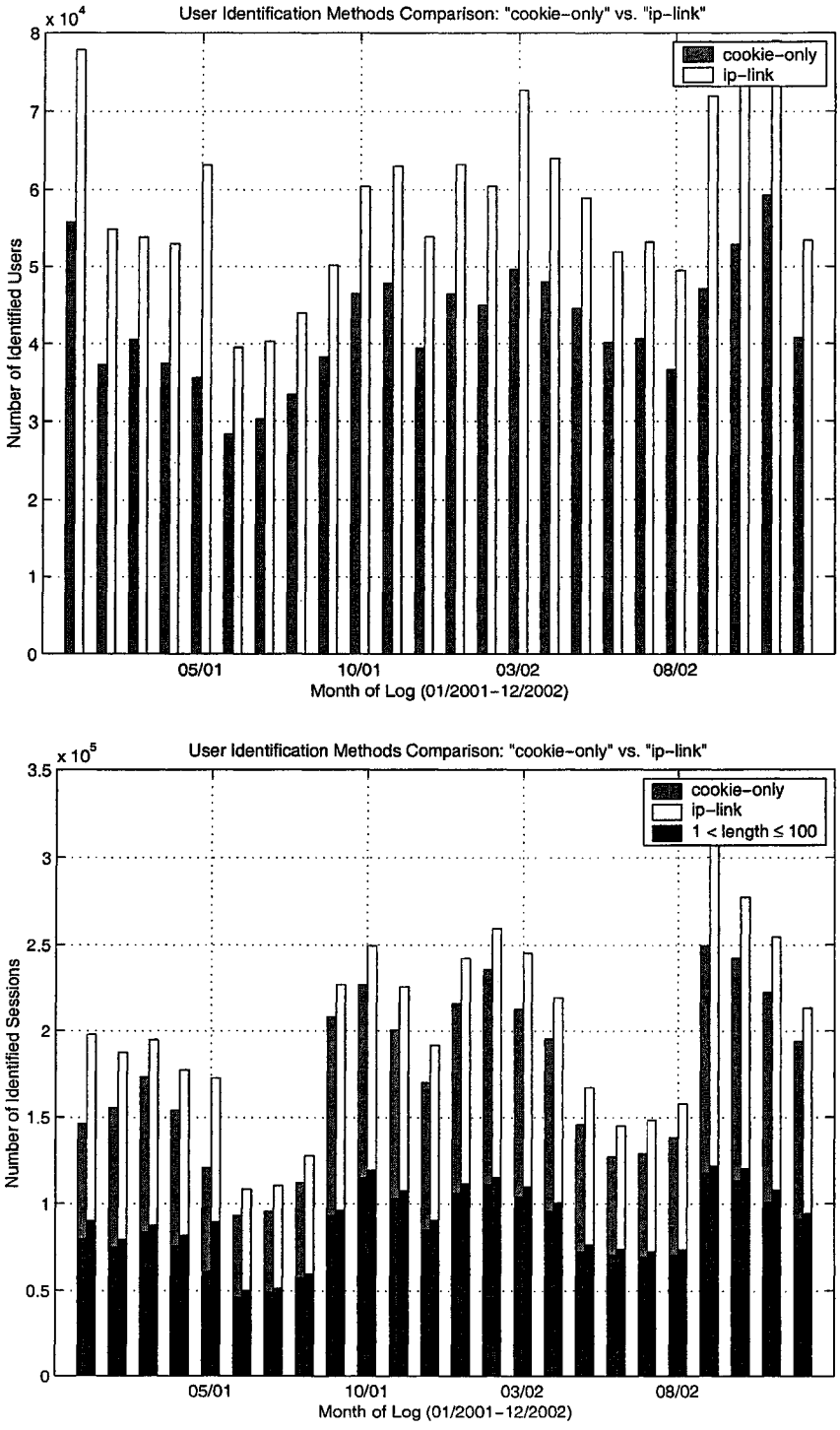


Figure 5.2: Experimental Results for User Identification Methods

Table 5.2: Experimental Results for Session Identification Methods

timeout-based vs. *link-based*

Term	Number of Sessions	Number of Sessions ($1 < length \leq 100$)	Average Session Length ($1 < length \leq 100$)
01/2001	146059 / 142511	79373 / 76208	5.73 / 5.94
02/2001	155300 / 150510	74970 / 70773	5.45 / 5.74
03/2001	173135 / 167992	83486 / 79050	5.29 / 5.55
04/2001	153813 / 148570	75687 / 70966	5.74 / 6.05
05/2001	120850 / 117355	61125 / 58172	5.97 / 6.21
06/2001	93123 / 90846	46225 / 44326	5.76 / 5.98
07/2001	95561 / 92997	49198 / 47032	5.76 / 6.00
08/2001	112402 / 108793	57011 / 53839	5.92 / 6.16
09/2001	208162 / 201777	93309 / 87625	5.69 / 6.00
10/2001	226526 / 218592	115384 / 108287	5.62 / 5.92
11/2001	200478 / 194048	103216 / 97574	5.37 / 5.63
12/2001	170163 / 165549	85046 / 81020	5.52 / 5.75
01/2002	215795 / 210148	106389 / 101448	5.62 / 5.85
02/2002	235623 / 229635	111424 / 106163	5.31 / 5.53
03/2002	212619 / 206044	104221 / 98472	5.32 / 5.60
04/2002	195312 / 188542	95746 / 89677	5.54 / 5.86
05/2002	145958 / 142246	72547 / 69343	5.94 / 6.15
06/2002	127465 / 124384	70389 / 67709	5.78 / 5.93
07/2002	129175 / 126153	69288 / 66755	5.92 / 6.10
08/2002	138355 / 135962	70443 / 68485	5.61 / 5.75
09/2002	249328 / 244361	118029 / 113868	5.16 / 5.30
10/2002	242430 / 236576	113755 / 108941	5.20 / 5.36
11/2002	222307 / 217122	100846 / 96542	5.11 / 5.29
12/2002	193534 / 188988	91200 / 87263	5.36 / 5.52

Note: Each value in this table is represented as $<V_t/V_l>$, where V_t is the result from the timeout-based method and V_l is the result from the link-based method.

difference in number of sessions is less than 7%, and the difference in average session length is less than 6%. This is illustrated in Figure 5.3.

In the following experiments of navigation improvement we arbitrarily choose to use only the timeout-based method for session identification. Based on the above results, it is believed that using the link-based method should not make a considerable difference.

5.3 Content Page Identification

Converting the access log data into user sessions is only the first step for data preparation. The next step is to determine content pages in each session. As previously discussed, content page is an important concept for describing users' "interest" in the web site, and our recommendation mechanism is designed to make recommendations only on these "interesting", or relevant pages.

But identifying content pages is notoriously difficult. Content pages can have different meanings in different environments. For those product-oriented e-commercial web sites, it is relatively easy to identify the content pages, i.e., interested products, based on the user's purchasing behaviors. And only those visits in which actual purchases were made are considered to be successful visits. However, for those general information-based web sites where no user feedback is involved, content page identification is not that straightforward, and can not be extracted from the user sessions in any very simple way. Under this circumstance, an ideal experiment should have the content pages pre-determined or manually assigned by the users. Otherwise, we can use a classification model to identify content pages from sessions, based on the meta-data and visiting history associated with each page. However, building such a model still requires a training dataset in which every content page has been pre-assigned. In our case, we want to determine content pages merely from the sessions without user interaction. Therefore, making use of certain heuristics is the only choice we have.

We have previously discussed two such heuristics for content page identification: one is Reference Length (RL), and another one is Maximal Forward Reference (MFR). Furthermore, we will propose a hybrid method called MFR-RL, which uses a combination of the heuristics from both MFR and RL. In what follows we discuss these three methods, and present the experiments we have conducted.

5.3.1 Content Page Identification Methods

5.3.1.1 Reference Length (RL)

The idea of the reference length heuristic is to identify content pages based on a cutoff viewing time. Here "reference length" is just another name for viewing time. This approach is derived from the assumption that each visited web page belongs to one of these two categories: *content page*, and *auxiliary page*, and any content page should have a longer viewing time than any auxiliary page. The problem with this approach is obvious: because users generally spend less time on those pages containing less information, these pages, even important to the users, might be missed as content pages. However, there is also the good part: the content pages

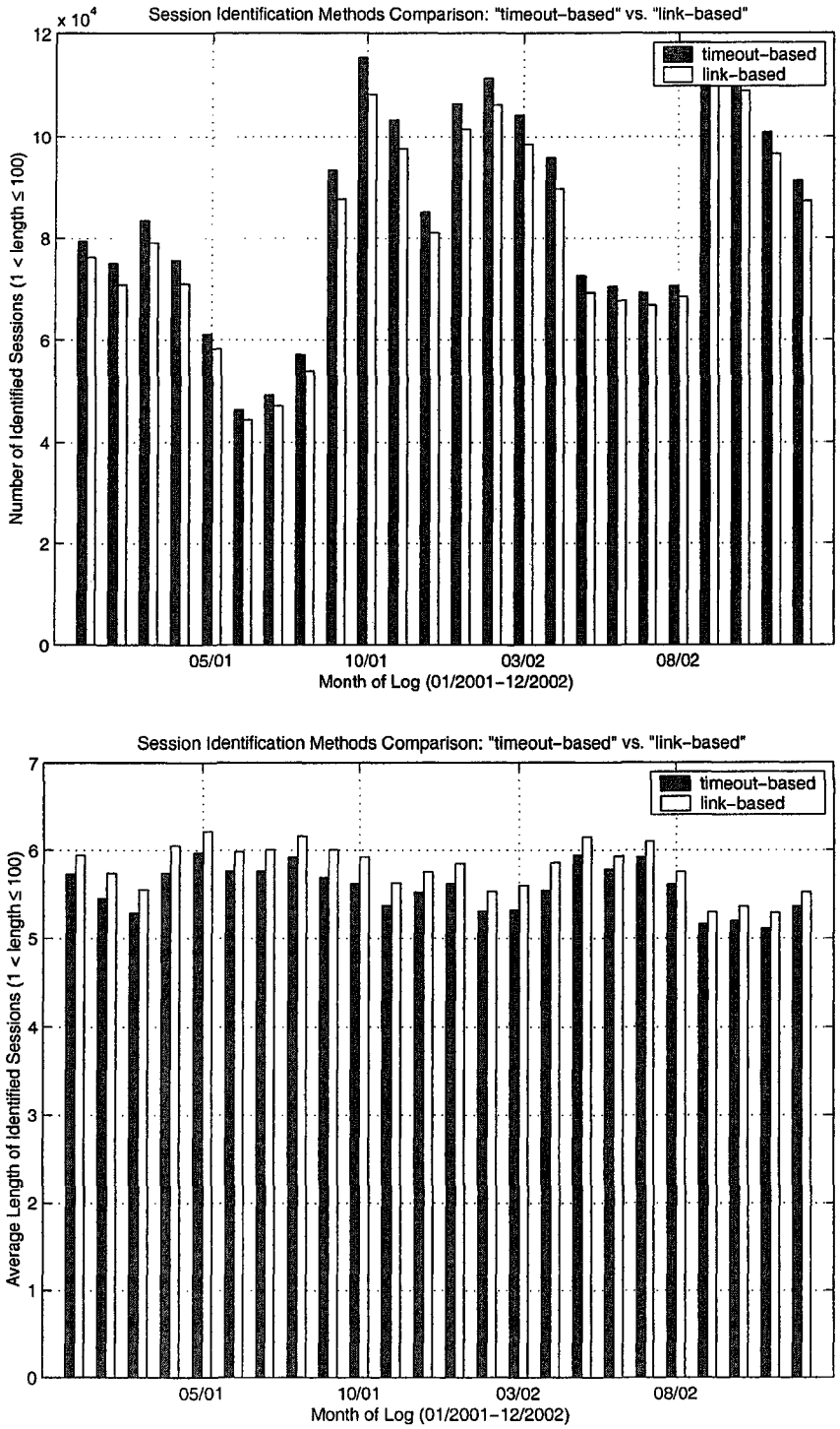


Figure 5.3: Experimental Results for Session Identification Methods

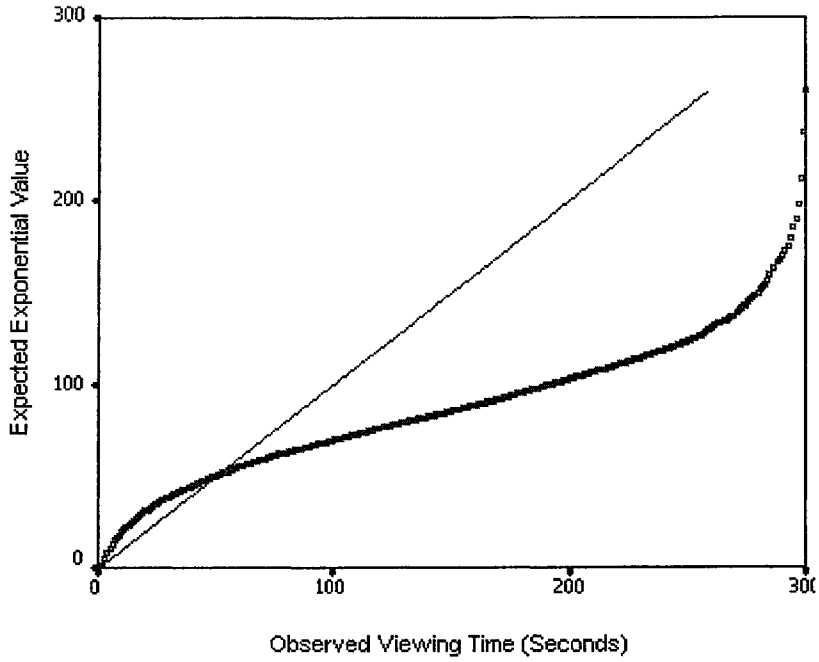


Figure 5.4: Exponential Q-Q Plot of Viewing Time (10/2002)

identified with this approach are highly probable to be real content pages, because users did spend relatively long time on them.

The cutoff viewing time can be set to an empirical value arbitrarily, e.g., 30 seconds or one minute. Also, Cooley et al. [21] proposed a method to compute this cutoff value from the distribution of the viewing time:

$$t = \frac{-\ln(1-\gamma)}{\lambda} \tag{5.1}$$

provided that the viewing time conforms to an exponential distribution, λ is the reciprocal of the observed mean viewing time, and γ is the estimated percentage of auxiliary pages among the entire log data under analysis. Here, λ can be calculated from the log data, but γ is still unknown and has to be assigned heuristically. However, our experiment showed that the viewing time does not necessarily conform to an exponential distribution, as illustrated by the Q-Q plot [13] in Figure 5.4. A diagram illustrating the viewing time distribution is given in Figure 5.5.

In this research we propose a new approach to solve this problem. The basic idea is that we can apply a clustering algorithm to the set of all viewing times to obtain two clusters: C_{short} and C_{long}, such that $\forall(c_1 \in C_{short}, c_2 \in C_{long}) : c_1 < c_2$. Then we can easily find a boundary that distinguishes C_{short} and C_{long}, and this boundary is the cutoff value we need.

In our experiments we use K-Means [45] as the clustering algorithm, and compute the boundary between C_{short} and C_{long} as:

$$t = \frac{mean_s + mean_l}{2}$$

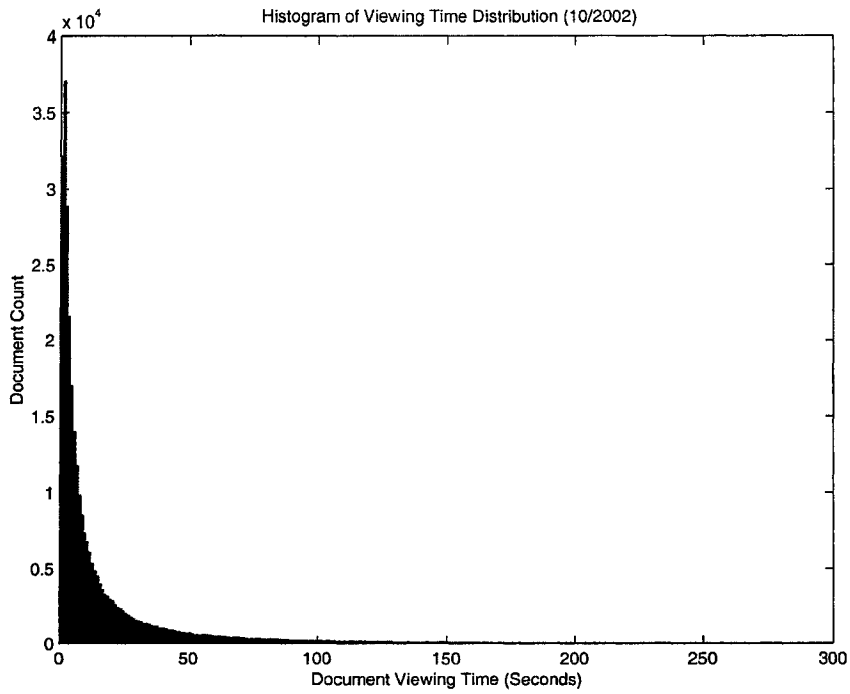


Figure 5.5: Viewing Time Distribution (10/2002)

where $mean_s$ is the mean viewing time in C_{short}, and $mean_l$ is the mean viewing time in C_{long}.

Some web pages may have very long viewing times, and these values can bias the clustering result enormously. To be more specific, these values can result in a much larger $mean_l$, which leads to a much larger cutoff value between C_{short} and C_{long}, therefore incorrectly cluster more values into C_{short}. To address this problem, in our experiments we simply exclude all the values greater than a given threshold. For example, suppose we know that a viewing time of one minute can determine a content page confidently, then in the clustering analysis we can exclude all the viewing times longer than one minute.

The experiments were conducted by calculating the cutoff viewing time for each monthly session data previously generated, excluding all viewing times longer than one minute. The experimental results are shown in Table 5.3.

The results show that the cutoff viewing times and the percentages of C_{short} obtained from different monthly data are quite consistent. The estimated cutoff viewing time is around 21.44 seconds ($\sigma = 0.77$), and the estimated percentage of auxiliary pages is around 79.90% ($\sigma = 2.12\%$). This is illustrated in Figure 5.6.

Another observation is that: if we compute the cutoff viewing time based on Equation (5.1) using $\lambda = \frac{1}{mean}$ and $\gamma =$"% of C_{short}" the result is also quite consistent with the cutoff viewing time obtained from our clustering algorithm. For example, the cutoff viewing time for October 2002 obtained from our clustering algorithm is 20.48 seconds. And by using Equation (5.1), given $mean = 11.01$ seconds and "% of C_{short}" = 82.89%, the estimated cutoff viewing time is 19.44 seconds. The difference between these two estimated values is only 5%.

Different from Equation (5.1), which requires an estimated percentage of auxil-

Table 5.3: Experimental Results for Cutoff Viewing Time Calculation

Term	Cutoff Viewing Time (*seconds*)	$mean_s$ (*seconds*)	$size_s$	$mean_l$ (*seconds*)	$size_l$	% of C_{short}
01/2001	22.04	7.34	153088	36.74	39027	79.69%
02/2002	20.93	6.63	146092	35.24	39028	78.92%
03/2001	20.87	6.41	158496	35.33	40478	79.66%
04/2001	21.51	6.81	160282	36.22	42316	79.11%
05/2001	21.81	7.48	125104	36.14	36655	77.34%
06/2001	22.46	7.95	91359	36.97	27040	77.16%
07/2001	23.18	8.45	88304	37.91	27728	76.10%
08/2001	23.14	8.34	111039	37.94	33848	76.64%
09/2001	21.41	6.82	200915	36.01	47970	80.73%
10/2001	20.71	6.26	246609	35.16	56825	81.27%
11/2001	21.34	6.72	203525	35.95	48257	80.83%
12/2001	21.41	6.92	176259	35.90	43068	80.36%
01/2002	21.46	7.12	224138	35.81	56171	79.96%
02/2002	21.25	6.63	202261	35.86	43954	82.15%
03/2002	20.87	6.68	211064	35.06	53361	79.82%
04/2002	21.35	6.94	195657	35.76	48772	80.05%
05/2002	21.75	7.53	149324	35.97	44100	77.20%
06/2002	21.65	7.41	127122	35.89	36485	77.70%
07/2002	22.18	7.66	148248	36.70	38622	79.33%
08/2002	21.49	7.21	164872	35.78	36371	81.93%
09/2002	20.25	5.63	253193	34.86	49448	83.66%
10/2002	20.48	6.08	232761	34.87	48038	82.89%
11/2002	20.48	6.03	210061	34.93	44044	82.67%
12/2002	20.57	6.09	182390	35.05	39023	82.38%

Note: In this table, $size_s$ is the size of C_{short}, and $size_l$ is the size of C_{long}. Moreover, "% of C_{short}" is computed as:

$$\frac{size_s}{size_s + size_l} \times 100\%$$

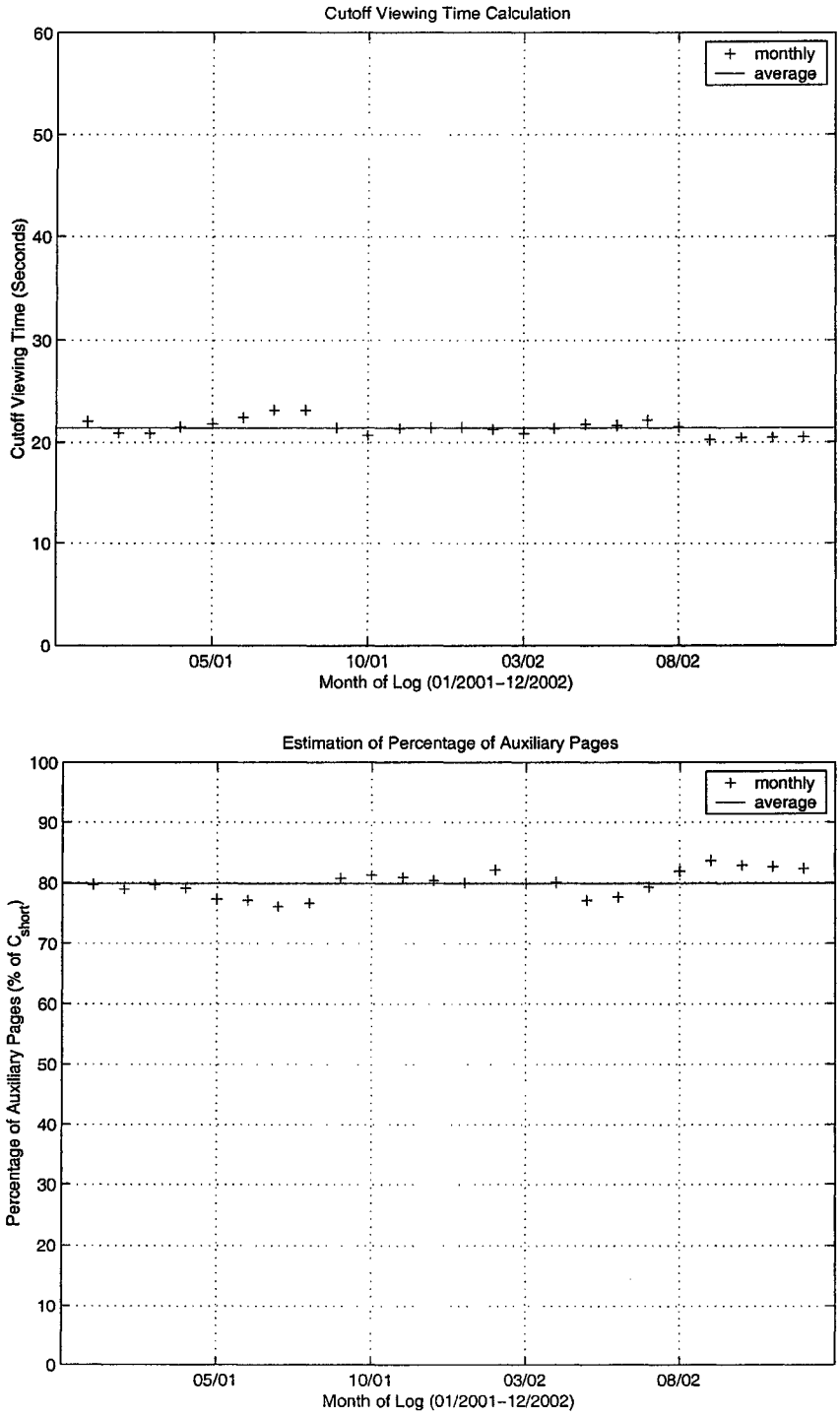


Figure 5.6: Experimental Results for Cutoff Viewing Time Calculation

iary pages, our approach only requires the user to specify an upper bound for viewing times used for clustering. The obtained cutoff viewing time changes monotonically based on this upper bound. While the basic idea of this upper bound is that it should be as small as possible without losing the ability to confidently determine a content page, how to find a proper value is still an open problem. In the above experiment, we simply set the upper bound to an arbitrary value — one minute.

In the following experiments of navigation improvement, we set the cutoff viewing time to 21.44 seconds, which is the mean cutoff viewing time of the above experiment.

5.3.1.2 Maximal Forward Reference (MFR)

The idea of the maximal forward reference heuristic is derived from the widely used hub-and-spoke structure throughout the Web [15]. It assumes that when the users find a content page, they usually move backward to the hub in order to reach other content pages. Therefore, the last page requested by a user before backtracking occurs is considered as a content page.

This method is considerably weak because its assumption, that any backtracking is because the user has found the content page, is excessively ideal and far from reality. There could be other reasons for backtracking as well. For example, it is also common that the user reaches a dead end or realizes being in the wrong way, and then backtracks to navigate in some other direction.

In our experiments we refine the MFR method by applying a minimum viewing time. The idea is that even if a page is the last page requested before a backtracking, it is still *not* a content page if the user has hardly spent any time to view it. Therefore, this minimum viewing time should be selected so that any requested page with a viewing time less than it can be confidently determined as an auxiliary page. So far this minimum viewing time can only be set arbitrarily, and in our experiments it was set to 5 seconds.

To experiment with this idea of improving the basic MFR method, we generated two sets of MFR-based content pages from the UACS log data, one using the basic MFR method, and another one using the refined MFR method. The results show that our refined MFR method can reduce the number of identified content pages by an average rate of 28.83% ($\sigma = 9.57\%$), and reduce the number of distinct content pages by an average rate of 25.19% ($\sigma = 14.70\%$). However, the reason why there is a large variance in the results is still unclear.

5.3.1.3 MFR-RL

MFR-RL is a hybrid method for content page identification. Its idea is to identify content pages using both heuristics from MFR and RL. With this method, a web page is a content page only if it is determined as a content page using both MFR and RL methods. It can be expected that MFR-RL has a smaller coverage (number of content pages identified) than both MFR and RL. However, due to its more strict requirement its quality could be higher.

MFR-RL-based content pages can be obtained by applying the refined MFR method with the minimum viewing time set to the cutoff viewing time of the RL method. As described earlier, in our experiments this cutoff viewing time is set to 21.44 seconds.

5.3.2 Significance of Content Pages

We can obtain a large number of content pages with each of the above methods. However, not all the obtained content pages are equally important. Some of them may not be true content pages at all. In this research we propose a number of criteria that can be used to further refine the results of above content page identification methods, and determine the relative importance of each content page obtained.

First, we define the *support* and *confidence* of a content page as follows: given a specific content page, its support is the number of times this page has been visited as a content page, and its confidence is the probability that this page was visited as a content page. Suppose a web page has been visited vc_s times totally, but only vc_c times as a content page, then the formal definitions of its support and confidence are:

$$support = vc_c \ , \ confidence = \frac{vc_c}{vc_s} \tag{5.2}$$

Another criterion is called *significance*, which can be used to determine the importance of the content page. The importance of the content page should be determined based on both the support and confidence criteria, and either of these two criteria being too small should dramatically decrease the importance. Therefore, we define the significance of a content page as the Geometric Mean of its support and confidence:

$$significance = \sqrt{support \times confidence} \tag{5.3}$$

The significance criterion is valuable only in a relative way, i.e., its absolute value does not carry any useful information. It is used only to compare the importance of different content pages. Moreover, the significances of content pages obtained from different identification methods are not comparable.

In the following experiments of navigation improvement, we can refine the content page identification results based on the above criteria, e.g., by eliminating content pages whose support and confidence values do not meet some pre-determined minimum thresholds. Apparently, higher support and confidence thresholds will result in a smaller but more strict (therefore, with potentially higher quality) set of content pages, while lower support and confidence thresholds will result in a larger set of content pages with potentially lower quality. An experiment of the relationship between these two thresholds and number of identified content pages, derived from the UACS log data of October 2002, is shown in Table 5.4.

Moreover, when the number of content pages (or content pages to be recommended) is restricted, we can select more important content pages based on their significance values.

5.3.3 Accuracy of Heuristically Identified Content Pages

So far the accuracy of heuristically identified content pages can only be evaluated by humans. But even human-identified content pages can still be inaccurate. As previously discussed, a page can be a content page for some users while being an auxiliary page for other users. Therefore, strictly speaking, only those content pages identified by the original users are guaranteed to be accurate. Unfortunately, having users identify content pages during their navigation is only applicable in certain experiments but infeasible in the real world.

Table 5.4: Refinement of Content Page Set (10/2002)

Minimum Support	Minimum Confidence	Number of Identified RL-based Content Pages
1	0%	13575
2	0%	7716
3	0%	5465
1	5%	12551
2	5%	7281
3	5%	5172
1	25%	6010
2	25%	3168
3	25%	2217

In our later experiments of different content page identification heuristics, we concentrate on the potential of those identified content pages in improving user navigation.

Chapter 6

Learning

We have discussed a number of learning techniques for web usage mining in Section 2.2.2. Since our goal is to improve user navigation through dynamic recommendations, what can be applied here are those techniques that can predict the user's "interests" given the current navigation path. In these experiments we explore four of them: association rule mining, sequential rule mining, usage-based clustering, and content-based clustering.

It is worth noting that no matter which learning technique we use and what form the learning results are represented in, the learning results are used for dynamic recommendations through a unique form — Navigation Compression Models.

6.1 Association Rule Mining

While the idea of association rule mining is to discover the co-occurrence patterns of different items in individual transactions, in the domain of web usage mining, each item is represented as a web page and each individual transaction is represented as a user session. The algorithm we used for association rule mining is DHP (Direct Hashing and Pruning) [54, 55], which was first proposed by Park et al. in 1995.

The general process of mining association rules between items in a set of transactions (e.g., the apriori algorithm [1]) is accomplished in two phases. Phase one is to obtain large itemsets, where each large itemset is a set of items with a sufficient support, i.e., the number of transactions in which the itemset appears is larger than a given threshold. Generally the large itemsets are discovered iteratively in the order of the itemset size, starting from itemsets with only one item. During each iteration, we first generate candidate (k+1)-itemsets from those large k-itemsets already obtained by joining any two large k-itemsets if they have exactly (k-1) items in common. Then, the occurrences of the candidate (k+1)-itemsets are counted throughout the transactions so that the actual large (k+1)-itemsets can be determined. Phase two is to generate association rules from the large itemsets obtained. For each large k-itemset ($k \geq 2$), we first obtain all its combinations of n-item subsets and (k-n)-item subsets, then make an association rule from each combination of such two subsets — one subset as the antecedent and another one as the consequent. Finally, all the association rules are checked for validity based on the *confidence* and *lift* thresholds (see Section 2.2.2.1).

The DHP algorithm achieves performance improvement in the phase of obtaining

```
;[support, confidence, lift]

(Rule A) [0.53%, 80.49%, 150.96]
"/iip"        --> "/iip/IIP/index/main.htm",
                  "/iip/IIP/index/menu.htm",
                  "/iip/IIP/index/blank.htm",
                  "/iip/IIP/index/bottom.htm",
                  "/iip/IIP/index/banner.htm"

(Rule B) [0.51%, 51.93%, 24.56]
"/programs/" --> "/programs/graduate"

(Rule C) [0.39%, 72.13%, 111.34]
"/~pawel/"   --> "/~pawel/COURSES/313/cmput313.html"
```

Figure 6.1: Association Rule Examples from UACS Log Data (10/2002)

large itemsets, by hashing the itemsets generated and pruning the space of candidate itemsets as well as the space of transactions along the mining process. In our implementation, this is done by adding the following processes into the basic algorithm:

- For each candidate (k+1)-itemset obtained using the basic algorithm, we can check if all its k-item subsets are large k-itemsets. If not, then this candidate can not be a large itemset and therefore can be removed before the counting process.
- During the phase of large itemset generation, we maintain an "active" itemset, which contains only those items that are still useful for the next iteration. After each iteration of generating the candidate itemsets, this "active" itemset is updated by removing those items not in the newly generated candidate itemsets. Then, the transactions can be pruned by removing items not in the "active" itemset.

When used for dynamic recommendations, an association rule is applied as follows: whenever the pages in the antecedent of the association rule have fully appeared in the user's current visited path, we recommend those pages in its consequent that have not been visited in the current path.

6.1.1 Interestingness of Association Rules

When the support and confidence are set to relatively low values, we can expect to obtain a large number of association rules. However, not all the association rules are useful. For example, Figure 6.1 shows three association rules extracted from the UACS log data in October 2002.

Rule A is actually generated from a page view (see page 18 for the definition of a page view). When we looked into page "/iip", we found that the page uses five frames within it, which are exactly those five pages in the consequent of Rule A. This means that whenever page "/iip" is requested, those five pages will be requested automatically by the client browser. Such being the case, Rule A is considered

completely useless in the application of navigation improvement. We have already processed page views in the data preparation process, by combining all the log records of the page view into a single request of the main page. However, there could be page views that have not been discovered automatically, and other kinds of page groups that we know for certain to be uninteresting.

Rule B is a different situation. When we looked into page "`/programs/`", we found that the page contains a list of hyperlinks to various information, including Undergraduate Programs ("`/programs/undergraduate`"), Graduate Programs ("`/programs/graduate`"), Industrial Internship Program ("`/iip`"), etc. Therefore, Rule B implies that most visitors who accessed page "`/programs/`" will be interested in the information of Graduate Programs. Rule B is not completely useless, but it is still not very useful in the effort of improving user navigation, because page "`/programs/`" already contains a hyperlink to the Graduate Programs, and more importantly, the hyperlink pointing to the Graduate Programs is already quite clear in the page that the visitors can easily locate it.

Different from rule A and rule B, rule C is considered reasonably useful. This rule indicates that most users visiting the homepage of professor Pawel Gburzynski ("`/~pawel/`") were actually interested in one of the courses he teaches — CMPUT 313 : Telecommunications and Computers ("`/~pawel/COURSES/313/cmput313.html`"). The difference between rule C and rule B is that in rule C there is no existing hyperlink between the antecedent and consequent. Therefore, if we can guide the user from the antecedent directly to the consequent, then the navigation pages in between will be saved.

Due to the potentially large number of the association rules obtained, removing useless rules manually is inefficient and not always practical. To deal with this problem, we introduced the following two methods, corresponding to the two different situations of rule A and rule B described above.

Uninteresting Page Cluster

In the first situation, we introduce a concept called *uninteresting page cluster*. Each uninteresting page cluster is defined as a page set such that any relationship between pages in the set is considered uninteresting. For example, suppose page A consists of two frames: page A_1 and page A_2. Then we know that any relationship between A, A_1, and A_2 should be uninteresting. Therefore, an uninteresting page cluster can be created to reflect this information: $C = \{A, A_1, A_2\}$. After that, in the process of association rule discovery, we can simply discard any large item set with multiple pages in C so that all the uninteresting rules related to C can be avoided. In this particular example, we can avoid twelve uninteresting rules by creating and using C:

$$\{A \rightarrow A_1\}\{A \rightarrow A_2\}\{A_1 \rightarrow A\}\{A_1 \rightarrow A_2\}\{A_2 \rightarrow A\}\{A_2 \rightarrow A_1\}$$

$$\{A \rightarrow A_1, A_2\}\{A_1 \rightarrow A, A_2\}\{A_2 \rightarrow A, A_1\}$$

$$\{A, A_1 \rightarrow A_2\}\{A, A_2 \rightarrow A_1\}\{A_1, A_2 \rightarrow A\}$$

Using uninteresting page clusters is much more efficient than removing corresponding uninteresting rules manually. Uninteresting page clusters can be defined by humans. Also, some uninteresting page clusters can be extracted from the log data automatically, e.g., the clusters regarding multi-frame page views.

Interestingness of Association Rules

In the second situation, we introduce another more typical measure for association rules, which is called *interestingness*. There has been a lot of research on determining the interestingness of association rules, which can be classified into two major categories: objective interestingness and subjective interestingness. Objective interestingness is data-driven, which measures the rule's interestingness based on the underlying data and the rule's general features such as rule structure and statistical consequence (e.g., support, confidence, and lift) [58, 23, 28]. Subjective interestingness is user-driven, and measures the rule's interestingness based on its relative value in a particular domain, which is often explicitly specified by the user through certain templates or specification language [70, 39, 42].

The interestingness defined here is a subjective measure, which is determined based on the web link structure: we simply assume that any association rule suggesting existing links is potentially uninteresting. Consequently, in our specific domain the interestingness of an association rule is a binary value: interestingness 1 indicates that the rule is potentially interesting, and interestingness 0 indicates that the rule is potentially uninteresting.

We acknowledge that page layout is also an important factor in this problem. Suppose a web page has a very bad layout, and the users can hardly find any useful information in it. Then an association rule suggesting something in this page is still useful. However, in our experiments we ignore page layout in the interestingness measure for simplicity.

Given an association rule

$$R = \{A \rightarrow B\} \quad (A = \{a_1, \cdots, a_m\}, B = \{b_1, \cdots, b_n\})$$

the interestingness of the rule is defined as follows:

- If $\exists \langle a_i, b_j \rangle$ $(a_i \in A, b_j \in B)$, there is a hyperlink in a_i linking to b_j, then $interestingness(R) = 0$.
- If $\forall \langle a_i, b_j \rangle$ $(a_i \in A, b_j \in B)$, there is no hyperlink in a_i linking to b_j, then $interestingness(R) = 1$.

Our definition of interestingness, though seemingly excessively strict, will not lose interesting rules. For example, suppose we have an association rule $\{a \rightarrow b, c\}$, and there exists a hyperlink from a to c, then this rule will be considered uninteresting and discarded. However, the useful information of this rule can still be captured by another rule: $\{a \rightarrow b\}$. From the association rule discovery process, we know that if we have $\{a \rightarrow b, c\}$, then we must have $\{a \rightarrow b\}$ and $\{a \rightarrow c\}$ as well.

6.1.2 Significance of Association Rules

We propose a significance criterion for association rules in order to determine their relative importance. Similar to the significance criterion of content pages, the importance of association rules should be determined based on both the support and confidence criteria, and either of these two criteria being too small should dramatically decrease the importance.[1] Therefore, the *significance* of an association rule is

[1]Here we exclude the lift measure of association rules in the definition of significance for simplicity. A more comprehensive definition of the significance (but not necessarily a better one) can also take lift into consideration.

also defined as the Geometric Mean of its support and confidence:

$$significance = \sqrt{support \times confidence} \tag{6.1}$$

When the number of association rules (or association rules to be used for recommendation) is restricted, we can select more important rules based on their significance values. Moreover, while there are multiple rules taking effects at the same time, significance can be used as one of the measures for determining the priorities of the rules.

6.1.3 Matching Order of Association Rules

The matching order of association rules is defined as follows:

> Given an association rule $R = \{A \rightarrow B\}$, its *matching order* is defined as length of the antecedent of the rule: $|A|$.

Matching order can also be used as one of the measures for determining the priorities of association rules, while there are multiple rules taking effects at the same time. Higher matching orders result in higher priorities.

6.1.4 Coverage of Association Rules

We define two kinds of coverages on sets of association rules as follows:

- Given a set of association rules, the *total coverage* of this rule set is defined as the number of distinct items included in these rules (i.e., in the antecedents or in the consequents).
- Given a set of association rules, the *consequent coverage* of this rule set is defined as the number of distinct items included in the consequents of these rules.

6.1.5 Redundancy of Association Rules

There are two different cases in which an association rule can be defined redundant:

- Given an association rule $R = \{A \rightarrow B\}$, if there exists another association rule $R' = \{A \rightarrow B'\}$ and $B \subset B'$, then the association rule $R = \{A \rightarrow B\}$ is considered redundant.
- Given an association rule $R = \{A \rightarrow B\}$, if there exists another association rule $R' = \{A' \rightarrow B\}$ and $A \supset A'$, then the association rule $R = \{A \rightarrow B\}$ is considered redundant.

Given the above definition, the first reaction is that any redundant association rule should be discarded. However, simply removing rule R is not an appropriate solution in our particular domain because rule R and rule R' may have different significance values, which means that they could have different priorities when used for recommendation.

Our solution to this redundancy problem is relatively simple: *keeping only those association rules with a single item in its consequent.* This solution reduces the redundancy of the first kind, but the second kind of redundancy has to be retained. The reasonability of this solution can be proved as follows:

Given an association rule

$$R = \{A \rightarrow B\} \qquad B = \{b_1, \cdots, b_n\}\ (n > 1)$$

Then it is easy to know that we must also have all the following rules

$$R_i = \{A \rightarrow b_i\} \qquad (i = 1, \cdots, n)$$

and for each R_i,

$$support(R_i) \geq support(R),\ confidence(R_i) \geq confidence(R)$$

and consequently,

$$significance(R_i) \geq significance(R)$$

It is apparent that the matching order and coverage of rule R are both preserved by the set of R_i's. Since rule R and R_i's have exactly the same antecedents, whenever rule R is applicable in the dynamic recommendation process, all R_i's are applicable as well. Therefore, each b_i recommended by rule R will also be recommended by the corresponding R_i, only the recommendations from R_i's always have higher priorities (because the significances of R_i's are higher than that of R; described in detail later in the evaluation section). However, while a particular document is recommended by multiple NCMs at the same time, only the highest priority value is used for its recommendation. Therefore, rule R will never be used.

6.1.6 Content-Page-Related Association Rules

In the primary application of association rule mining, the market basket analysis problem [2], the meaning of an item occurring in a transaction is simple and unique: that item was purchased by the consumer in that specific transaction. However in the domain of web navigation, a web page occurring in a session can have many different meanings. From the view point of a page's relevance to the user, a web page occurring in a session can have at least two different meanings: (1) that page was visited by the user as a content page in that specific session, or (2) that page was visited only as an auxiliary page in the session.

Because a web page may stand for different meanings under different circumstances, it can be expected that dealing with each meaning on its merits will result in different kinds of association rules. For example, given b as a particular web page and b_c being the same page only requested as a content page, $R = \{a \rightarrow b\}$ and $R' = \{a \rightarrow b_c\}$ are actually two different rules. Provided that page a was visited in a specific session, rule R implies that page b was probably visited in the same session as well, while rule R' suggests that page b was probably also visited in the same session *as a content page*.

This idea is especially useful in our experiments of improving user navigation through dynamic recommendations. As mentioned earlier, we recommend only content pages to the user. Such being the case, by treating content pages and auxiliary

Table 6.1: Content-Page-Based Classification of Association Rules

Type of Rules	*Description*
content-page-insensitive	Content pages and auxiliary pages are not distinguished in association rule mining, i.e., $p_c = p_{nc}$.
content-page-oriented	Content pages and auxiliary pages are treated differently in association rule mining, i.e., $p_c \neq p_{nc}$; and only content pages can appear in the consequents of the rules.
content-page-only	Content pages and auxiliary pages are treated differently in association rule mining, i.e., $p_c \neq p_{nc}$; and only content pages can appear in the rules.

p_c : page p as a content page
p_{nc} : page p as a non-content (auxiliary) page

pages differently, we might be able to avoid a significant number of rules which will never be applied. Still, using the last example, let b_c be the content page b and b_{nc} be the non-content (auxiliary) page b. By treating b_c and b_{nc} differently, instead of getting $R = \{a \rightarrow b\}$, what we might get is $R_1 = \{a \rightarrow b_c\}$ and $R_2 = \{a \rightarrow b_{nc}\}$. And we know that R_2 will never be used, therefore can be discarded, because b_{nc} is not a content page.

Based on their sensitivities to content pages and the ways content pages are employed, association rules can be classified into three major categories: *content-page-insensitive*, *content-page-oriented*, and *content-page-only*, as shown in Table 6.1.

6.2 Sequential Rule Mining

The algorithm we used for sequential rule mining is a modified version of DHP (Direct Hashing and Pruning) [54, 55], in which the ordering information in user sessions is taken into account. In this modified DHP, all the itemsets and transactions are ordered, i.e., an itemset $<a, b>$ means that page a must be requested before page b in user sessions. However, adjacent items in the itemsets are not necessarily adjacent in the transactions. For example, a transaction $\{a, b, c\}$ contains three 2-itemsets: $<a, b>$, $<b, c>$, and $<a, c>$, where $<a, c>$ is not adjacent in the transaction. Moreover, there could be duplicate items in one itemset, although adjacent duplicates are not allowed.

A sequential rule is applied in the similar way an association rule is applied for dynamic recommendations, except that the pages in the antecedent of the sequential rule must appear in the same sequence in the current path, as they appear in the sequential rule itself. For example, given a current user path $\{\cdots, b, \cdots, a\}$, an association rule $\{a, b \rightarrow c\}$ will recommend page c, but a sequential rule $\{a, b \rightarrow d\}$ will not make any recommendation because the sequence between a and b (a requested before b) is not satisfied.

Similar to association rules, sequential rules can also be measured by interestingness and significance, also have the features of matching order and coverage, and also

have the redundancy problem which is dealt with in the same way as with association rules. Moreover, sequential rules can also be classified into the three content-page-based categories as defined above in Table 6.1: *content-page-insensitive*, *content-page-oriented*, and *content-page-only*.

6.3 Clustering

We are also interested in how we can apply some form of clustering to improve user's web navigation. Here we used two different approaches for clustering web pages. One is usage-based, and another one is content-based. A web page cluster is applied for dynamic recommendations as follows: if one or more pages in the cluster have been visited in the user's current path, we recommend other unvisited pages in the cluster or a limited number of unvisited pages that have higher priorities (based on certain measures).

6.3.1 Usage-based Clustering

The algorithm we used for usage-based clustering is PageGather [56]. This algorithm finds purely usage-based page clusters based on the co-occurrence frequencies between web pages. The procedure of the original PageGather algorithm is described as follows:

1. Process server access logs into user sessions.
2. Generate a similarity matrix from the co-occurrence patterns between web pages. The similarity between two pages p_1 and p_2 is computed as:

$$sim(p_1, p_2) = \begin{cases} \min\{P(p_2 \mid p_1), P(p_1 \mid p_2)\}, & \text{if } p_1 \text{ and } p_2 \text{ are unlinked} \\ 0, & \text{if } p_1 \text{ and } p_2 \text{ are linked} \end{cases} \quad (6.2)$$

   where $P(p_2 \mid p_1)$ and $P(p_1 \mid p_2)$ correspond to the confidence measure in association rule mining.
3. Create an undirected and unweighted graph based on the similarity matrix. Each node represents a web page, and each edge indicates that the similarity between the connected two pages is greater than a given threshold.
4. Find cliques or connected components in the graph.
5. Create a page cluster for each clique or connected component found.

As previously discussed, connected components will result in much larger but less useful clusters. Therefore, in our experiments we focused only on the discovery of cliques.

The original PageGather algorithm has two problems. First, this algorithm may find occasional page co-occurrence patterns. For example, suppose page A and page B both appear only once in the logs, and they appear together in the same user session, then $\{A, B\}$ will be identified as a page cluster because $P(B \mid A) = 1$ and $P(A \mid B) = 1$. Such being the case, the number of page clusters found by this algorithm could be huge. Second, there is no cluster center, and there is no weight indicating the distance between cluster nodes.

To solve these problems, we made a few modifications to the original PageGather algorithm:

- A *support* threshold is introduced to further restrict the number of edges in the graph. This is similar to the support measure in association rule mining: here the support of a page pair $\{A, B\}$ is the probability (based on statistical result) that A and B occur in the same session:

$$support\{A, B\} = \frac{\text{number of sessions in which } A \text{ and } B \text{ both occur}}{\text{number of total sessions}}$$

  As noted earlier, an undirected and unweighted graph is created based on the similarity matrix, where each edge indicates that the similarity between the connected two pages is greater than a given threshold. In addition to this similarity threshold, here we also require that the support of the two connected nodes (as a page pair) is greater than some threshold.
  By applying the support threshold we can avoid occasional co-occurrence patterns, therefore considerably reduce the number and size of page clusters found. For example, when we applied the PageGather algorithm to the log data of October 2002 with the similarity threshold set to 50% and no minimum support, we found 7522 page clusters and the maximum cluster size was 66. But when we set the support to 0.005%, which means each page pair in the same cluster should have occurred together in the same session at least 5 times, the number of clusters found was reduced to 2798 and the maximum cluster size was reduced to 42.
- When a page cluster is found, we also record the similarity between each page pair in it. The clique-finding algorithm we used was proposed by Bron et al. [6] in 1973. This algorithm finds cliques in an undirected and unweighted graph. We modified this algorithm so that it can handle edge weights. Every edge in the graph has a weight associated with it, and the weight is the similarity between the two connecting nodes. Therefore, the results of our clique-finding algorithm include not only clusters of pages, but also the similarity between every two connecting nodes which indicates how closely the corresponding pair of web pages are related to each other.
  The similarity between page pairs can be used for determining the priorities of cluster-based recommended pages. For example, suppose page A has been visited in the user's current path, page B and C are recommended because they are in the same cluster with page A, and $sim(A, B) > sim(A, C)$, then page B should have higher priority than page C.

6.3.1.1 "Interesting" Page Clusters

The original PageGather algorithm tries to discover related (based on co-occurrence patterns) but unlinked pages: if two pages p_1 and p_2 are linked, then $sim(p_1, p_2) = 0$. This idea is similar to the concept of "interestingness" we defined for association rules. Therefore, here we call those page clusters obtained using this algorithm "interesting" page clusters.

However, this idea of "interestingness" is not guaranteed valuable on the navigation improvement problem. Therefore, we can generalize the PageGather algorithm to allow linked pages appearing in the same cluster. This is done by eliminating the hyperlink-based restriction in the definition of similarity:

$$sim(p_1, p_2) = \min\{P(p_2 \mid p_1), P(p_1 \mid p_2)\} \tag{6.3}$$

6.3.1.2 Significance of Page Clusters

Here we propose a significance criterion for page clusters in order to determine their relative importance. The *significance* of a page cluster is defined as the average similarity between any page pair in the cluster:

$$significance(C) = \frac{\sum_{p_1 \in C, p_2 \in C} sim(p_1, p_2)}{|\langle p_1, p_2 \rangle|}$$

where C is a page cluster, p_1 and p_2 are two different pages in the cluster, and $|\langle p_1, p_2 \rangle| = \begin{pmatrix} |C| \\ 2 \end{pmatrix}$ is the number of such page pairs.

When the number of page clusters (or page clusters to be used for recommendation) is restricted, we can select more important clusters based on their significance values.

6.3.1.3 Matching Order of Page Clusters

Similarity between page pairs is not the only measure that can be used for determining the priorities of cluster-based recommended pages. Matching order is another possible measure that can be used for this purpose.

> Given a page cluster C and a user navigation path P, the *matching order* of the cluster and the path is defined as the number of distinct pages they have in common: $|C \bigcap P|$.

The basic idea is that: higher matching orders mean better matches, therefore result in higher priorities. For example, given a user navigation path P, and two clusters C_1 and C_2, if $|C_1 \bigcap P| > |C_2 \bigcap P|$, then those recommended pages obtained from C_1 should have higher priorities than those obtained from C_2.

6.3.1.4 Coverage of Page Clusters

We define coverage on sets of page clusters as follows:

> Given a set of page clusters, the *coverage* of this cluster set is defined as the number of distinct items included in these clusters.

6.3.1.5 Content-Page-Related Page Clusters

As previously discussed, content pages play an important role in the learning process as well as the process of dynamic recommendation. Similar to association rules and sequential rules, page clusters can also be classified into three categories based on their sensitivities to content pages, as shown in Table 6.2.

6.3.2 Content-based Clustering

The algorithm we used for content-based clustering is the conceptual cluster mining algorithm proposed in [57]. The original conceptual cluster mining algorithm is fulfilled in three steps:

Table 6.2: Content-Page-Based Classification of Page Clusters

Type of Clusters	*Description*
content-page-insensitive	Content pages and auxiliary pages are not distinguished in cluster mining, i.e., $p_c = p_{nc}$.
content-page-oriented	Content pages and auxiliary pages are treated differently in cluster mining, i.e., $p_c \neq p_{nc}$; and each cluster should contain at least one content page.
content-page-only	Content pages and auxiliary pages are treated differently in cluster mining, i.e., $p_c \neq p_{nc}$; and only content pages can appear in the clusters.

p_c : page p as a content page
p_{nc} : page p as a non-content (auxiliary) page

1. Run cluster mining algorithm Ω on the document collection D to obtain a set C of usage-based clusters.
2. For each cluster c in C:
   (a) Run concept learning algorithm Γ to find the concept $v = \Gamma(c, D - c, L)$, where c is used as the positive example set, $(D - c)$ is used as the negative example set, and L is a conceptual language which is used to provide a description for each document in D.
   (b) Find the set c_v which is the extension of v in D.
3. Return all the sets c_v found.

The original conceptual learning algorithm has a significant limitation that it needs a pre-defined conceptual language and a pre-provided description for each page. This is indeed a very strong requirement. In our experiment, an alternative approach was employed: for each cluster $c \in C$ found by PageGather, its closest content-based cluster c_v is discovered. The modified conceptual cluster mining algorithm is described as follows:

1. Run cluster mining algorithm Ω on the document collection D to obtain a set C of usage-based clusters. In our experiment the PageGather algorithm is applied as Ω.
2. Run content-based document clustering algorithm Θ on the same document collection D to obtain a set C' of content-based (in our case, text-based) clusters. In our experiment, the CBC (Clustering By Committee) algorithm [53] is applied as Θ.
3. For each cluster c in C, find its most similar cluster c' in C'. Here a minimum similarity ε is applied to control the matching quality: if there is no c' in C' such that $similarity(c, c') \geq \varepsilon$, then no c' is returned for c.
4. Return all the clusters c' found.

Similar to usage-based page clusters, content-based page clusters also have the measures of matching order and coverage, which are defined in the same ways as those

of usage-based clusters. However, the significance of content-based page clusters is defined differently from that of usage-based clusters:

$$significance(C') = \frac{\sum_{p \in C'} sim(p, centroid(C'))}{|C'|}$$

where C' is a content-based page cluster, p is any page in C', $centroid(C')$ is the centroid of C', and $|C'|$ is the size of C'.

Moreover, since content-based page clusters are obtained purely from document text, all such clusters are content-page-insensitive.

6.3.2.1 CBC Algorithm

CBC (Clustering By Committee) [53] is a committee-based clustering algorithm. While traditional clustering algorithms represent the centroid of a cluster by averaging the feature vectors of all its elements (e.g., K-means) or by a single representative element (e.g., K-medoids), CBC represents the centroid by averaging only a subset of the cluster members, which is called a committee.

In CBC, each element is represented as a feature vector:

$$MI(e) = (mi_{ef_1}, mi_{ef_2}, \cdots, mi_{ef_n})$$

in which each feature value is defined as the mutual information between the element e and the feature f:

$$mi_{ef} = \log \frac{\frac{c_{ef}}{N}}{\frac{\sum_i c_{if}}{N} \times \frac{\sum_j c_{ej}}{N}}$$

Here, n is the number of features, c_{ef} is the frequency count of feature f occurring in element e, and $N = \sum_i \sum_j c_{ij}$ is the total frequency count of all features in all elements.

Then, the similarity between two elements e_i and e_j can be computed using the *cosine coefficient* of their mutual information vectors:

$$sim(e_i, e_j) = \frac{\sum_f mi_{e_i f} \times mi_{e_j f}}{\sqrt{\sum_f mi_{e_i f}^2 \times \sum mi_{e_j f}^2}}$$

The CBC algorithm consists of three steps:

1. Find each element's top-k similar elements (e.g., $k = 20$).
2. Construct a set of tight clusters, each of which forms a committee. The similarity between each pair of committees should be lower than a given threshold, and the union of all the committees is generally just a small subset of the entire document collection. Each committee defines one of the final clusters.
3. Each element is assigned to the cluster of the committee that it is most similar to.

6.3.2.2 Matching of Usage-based Clusters and Content-based Clusters

While searching the most similar content-based cluster for a usage-based cluster, what's important is the coverage of common pages, i.e., the number of pages included in both clusters. Therefore, to measure the similarities between content-based clusters and usage-based clusters, we only consider page occurrence in the clusters. All the other information, e.g., pair-wise page similarity or page similarity to the centroid of the cluster, is not used for this purpose.

Given a content-based cluster c_c and a usage-based cluster c_u, suppose c_c is the true result set and c_u is the result of information retrieval, we can define precision and recall as follows:

$$precision = \frac{|c_c \cap c_u|}{|c_u|}\ ,\ recall = \frac{|c_c \cap c_u|}{|c_c|}$$

where $|c|$ is the size (number of pages) of cluster c.

The similarity between c_c and c_u should be determined based on both the precision and recall criteria, and either of these two criteria being too small should dramatically decrease the similarity. For this purpose, we compute this similarity using F-measure (the Harmonic Mean of precision and recall) [63].

Given the precision p and recall r, the F-measure is defined as:

$$F = \frac{2}{\frac{1}{p} + \frac{1}{r}} = \frac{2 \cdot p \cdot r}{p + r}$$

Therefore,

$$similarity(c_c, c_u) = \frac{2 \cdot |c_c \cap c_u|}{|c_c| + |c_u|}$$

Chapter 7

Evaluation

While our ultimate goal is to improve user navigation, it can not be assessed without objective evaluation. The role of evaluation is to measure the efficiency of navigation before and after learning, in order to provide a basis for assessing the value of various data and learning combinations.

7.1 Evaluation Function

Measuring the efficiency of navigation is not a straightforward task. While this measurement can be perceived by humans through various kinds of visualizations [49, 50], we also require some quantitative measures which can be obtained automatically from the navigation trails with and without the help of dynamic recommendations. As noted earlier, these quantitative measurements are obtained by comparing the original and compressed sessions.

One possible approach to the development of such quantitative measurements is based on the number of hyperlinks traversed. The fewer hyperlinks traversed in the session, the more efficient the user navigation is. There could be other solutions as well, such as the time spent or the size of documents accessed, in a session. However, the time and size could vary a lot depending on different kinds of web contents and different user browsing preferences. Therefore, in our experiments we use only the number of traversal links for the quantitative measurements.

While the number of traversal links can indicate the efficiency of user navigation, we have to make sure that the compressed sessions keep all those "relevant" pages (also called content pages) in the corresponding original sessions so that the compression actually makes sense. Therefore, the definition of content pages plays an important role in the quantitative measurements mentioned above. While we use different content page sets obtained from various content page identification methods, the compressed sessions and the measured improvements of user navigation will be different as well.

The specific measure we propose here is called *Navigation Improvement* (NI), which indicates a quantitative improvement in the navigation efficiency of the compressed sessions over the original ones. Intuitively, NI can be computed as:

$$NI = \frac{N_{org} - N_{com}}{N_{org}} \tag{7.1}$$

where N_{org} is the number of traversal links in the original sessions, and N_{com} is the number of traversal links in the compressed sessions.

For example, suppose we have obtained an association rule $\{A \rightarrow D\}$, where D is a content page. Then an original session $S_{org} = \{A, B, C, D, E\}$, where B and C are auxiliary pages, can be compressed by removing B and C to obtain $S_{com} = \{A, D, E\}$. The navigation improvement for this session would be:

$$NI(S) = \frac{N_{S_{org}} - N_{S_{com}}}{N_{S_{org}}} = \frac{5-3}{5} = 40\%$$

However, this simple measure does not take the cost of dynamic recommendations into account. In an extreme example, suppose the user is provided with one hundred recommendations, and only one of them is useful in guiding the user to the next content page by skipping one auxiliary page. Such being the case, most users will probably consider this recommendation set useless because it will take them less effort simply browsing to the content page with two clicks than picking it up from the huge recommendation list.

So another quantitative method could compute improvement by estimating the cost of the recommendations and subtracting it from the NI. The basic idea is that the more recommendations we provide to the user, the more cost we should subtract from the NI. In our experiments, we define a specific parameter to represent the cost of one recommendation, which is denoted as r_cost. So $\frac{1}{r\_cost}$ recommendations will cancel out the benefit of one saved traversal link.

Note that the determination of r_cost actually reflects the user's (or web designer's) consideration on the tradeoff between size of the recommendation space and proportion of "interesting" information contained in that space. A larger recommendation space generally contains more "interesting" information, but also introduces much more "uninteresting" information (or noise) into that space. Therefore, a higher setting of r_cost indicates that lower noise level is preferred, and consequently, it is relatively more important to obtain a higher precision of recommendations rather than a higher recall. On the contrary, a lower setting of r_cost indicates that amount of "interesting" information is a larger concern than the noise level, and consequently, it is relatively more valuable to obtain a higher recall of recommendations rather than a higher precision.

The choice of the r_cost parameter will inevitably affect the evaluation results of cost-based navigation improvement, and accordingly affect the comparisons between different kinds of NCMs on their potentials in such improvement. However, the appropriate choice of this parameter should be determined by the status of the website and user preferences, and so far can only be obtained empirically.

Based on the above idea, a cost-based Navigation Improvement can be computed as:

$$NI_c = \frac{N_{org} - N_{com} - R\_cost}{N_{org}} \tag{7.2}$$

where $R\_cost = n_r \times r\_cost$, and n_r is the number of recommendations provided to the user during the session.

Note that NI_c is always smaller than, if not equal to, its corresponding NI. In the above example, given $n_r = 4$ and $r\_cost = 0.1$, which means that during the session there were four recommendations provided to the user, and ten recommendations

will cancel out the benefit of one saved traversal link, then the cost-based navigation improvement of the session would be:

$$NI_c(S) = \frac{N_{S_{org}} - N_{S_{com}} - n_r \times r\_cost}{N_{S_{org}}} = \frac{5 - 3 - 4 \times 0.1}{5} = 32\%$$

7.2 Navigation Compression Mechanism

Given the NCMs obtained from the learning process and the evaluation methods for measuring the navigation improvement, the only problem left in our experiments is how to obtain the compressed sessions from the original ones.

Our experiment is designed to obtain the compressed sessions by simulating dynamic-recommendation-engaged user navigation behavior on the original sessions, instead of collecting data from the real web usage. This approach requires that we address two issues: (1) the dynamic recommendation mechanism, and (2) the simulation of user actions on recommendations.

The dynamic recommendation mechanism determines how the recommendations are generated and presented to the user. Our recommendations are generated by applying NCMs obtained from the learning process to the user's current path. Therefore, while the user navigates in the web site, the user path changes continuously, and the recommendations are updated along the way. This is why we call the recommendations "dynamic."

How recommendations are presented to the user is also an important issue. First, the number of recommended hyperlinks must be limited. As previously discussed, finding the useful page in a huge recommendation list could be more difficult and time-consuming than simply navigating without any recommendation. For this purpose, in our experiments a specific parameter is defined to constrain the maximum number of recommendations that can be presented to the user in one screen, which is denoted as r_limit. So at any moment the user can only see up to r_limit most important recommendations.

Since we want to display the most important recommendations at the top of the recommendation list, the second issue of the dynamic recommendation mechanism is to determine the significance, or priority, of each recommended hyperlink. In our experiments the priorities of recommendations are determined based on the following criteria, which are checked in order:

1. The exact position in the user path where the recommendation is generated. More recently generated recommendations are given higher priority. This is because we believe that in most cases, the user interest at a specific point of time is more highly related to the documents visited close to that point. This is called the *Recentness Effect.*
2. The quality of the matching between the current user path and the NCM used to generate the recommendation. This matching quality can be defined variously for different kinds of NCMs. In our experiments, for rule-based (association rule and sequential rule) NCMs, the matching quality is defined as the matching order of the rule. For cluster-based NCMs, the matching quality is defined as the matching order between the cluster and the path. Recommendations generated by NCMs with higher matching quality are given higher

priorities. This determination is based on our assumption that the more information used for learning, the more accurate results we can obtain. This is called the *Better Matching Effect.*

3. The significance of the generated recommendation. This significance can also be defined variously based on different NCMs. In our experiments with rule-based NCMs, this significance is defined as the significance of the rule applied. With usage-cluster-based NCMs, this significance is defined as the similarity between the recommended document and the document (in the cluster applied) used to generate it. With content-cluster-based NCMs, this significance is defined as the similarity between the recommended document and the centroid of the cluster used to generate it. Recommendations with higher significance values are considered more useful, therefore given higher priorities. This is called the *Usefulness Effect.*

A particular document can be recommended simultaneously by different NCMs or at different positions in the user path. In such cases, only the one with highest priority is kept for further recommendation.

The user action model we used is simple, as it assumes that the user will follow all the "correct" recommendations. Here "correct" means that the recommended page is a content page to be visited in the session, and there is no other content page between the current position and the recommended page. Note that this "simulation" of a user's "correct" behavior for our experiments is really just a set of constraints on how to measure the value of the NCMs, and doesn't rely on or even prescribe a particular user behavior to be valid.

Here content pages play two important roles in the process of dynamic recommendation and navigation compression: (1) they are used to restrict recommendations, and (2) they are also used to determine the "correct" user navigation behaviors. However, in some cases (e.g., the UACS data set used in our experiments) content pages are not fixed and can only be determined using certain heuristics or learning techniques. Therefore, it can be expected that the content pages used for these two purposes can be different because they are obtained from different web log data. The content pages used for recommendations are obtained from the training data set, i.e., the same data set used to learn NCMs. The content pages used for simulating user behaviors and consequently achieving navigation compression are obtained from the test data set, i.e., the data set used to test the performance of applying NCMs. Using the same data set for both training and test can also be valuable, as it may to some extent reveal the best potential improvement that can be achieved on that data set.

As previously discussed, when the content page set is not fixed, we can guess at individual content pages using various heuristics instead of more intrusive data gathering (e.g., users' manual designation). However, using heuristics for content page identification is not guaranteed and so far there is no method that can evaluate the quality of the content pages obtained this way without human participation. Our evaluation for navigation improvement does not address or help address the problem of content page evaluation. But this does not mean that navigation improvement evaluation is meaningless without an accurate evaluation of the content pages. In fact, this uncertainty exists not merely in the content page identification problem, but throughout the entire web mining process. For example, we also use various heuristics in the data preparation process for user identification, session identifica-

Original Session : 1, 2, (3), 4, (5), 6, (7), 8, (9), 10
Recommendation Content Page Set : (5), (7), (9)
Test Content Page Set : (3), (5), (7), (9)
NCM Set : $1 \rightarrow 9$
$2 \rightarrow 5$
$1,2 \rightarrow 4,7$
$1,3 \rightarrow 9$ $(significance = 0.5)$
$2,3 \rightarrow 5$ $(significance = 0.8)$

Navigation Path	Recommendation List
1	(9)
1, 2	(7), (5), (9)
1, 2, (3)	(5), (9), (7)
1, 2, (3), (5)	(9), (7)
1, 2, (3), (5), (7)	(9)
1, 2, (3), (5), (7), (9)	null
1, 2, (3), (5), (7), (9), 10	null

Compressed Session : 1, 2, (3), (5), (7), (9), 10

Figure 7.1: Navigation Compression Example

tion, and page view identification. Therefore, while there exist different heuristics with corresponding evaluation methods in different phases of the web mining process, navigation improvement evaluation is the final step which is used to determine the potential improvement in user navigation provided with the data and learning results obtained from a combination of heuristics in different phases.

An example is shown in Figure 7.1, which illustrates how our navigation compression mechanism works. This example uses sequential-rule-based NCMs, and each page is represented as a page ID, while content page IDs are enclosed with "()"s.

7.3 Experiments

Here we report experiments with four kinds of NCMs, including NCMs obtained from association rules, sequential rules, usage-based page clusters, and content-based page clusters. For each kind of NCM, we tested the navigation improvement with a variety of parameter settings. And different kinds of NCMs were compared based on their performance on improving user navigation.

7.3.1 Training Data and Test Data

As previously described, the web content changes rapidly, and user interest may change constantly as well. Such being the case, it can be expected that the effectiveness of the knowledge learned from the previous users may decrease continuously over time. Therefore, it is necessary for the knowledge base to be updated regularly. In our experiments, we used the UACS server access logs and assumed that the knowledge base (which is composed of NCMs) is updated on a monthly or weekly

basis. In the case of monthly updating, we used the previous one-month logs as training set, and the following one-month logs as test set. In the case of weekly updating, we used the previous one-month logs as training set, and the following one-week logs as test set.

For those NCMs obtained from association rules, sequential rules, and usage-based clusters, the experiments can be conducted on web logs from any time period. However, the situation is different for NCMs obtained from content-based clusters. Since web content changes continuously, to generate content-based clusters it is preferred to use web log data and web content data collected from the same time period. For example, it may not be appropriate to use the web content data collected at December 2002 with the web log data of June 2002 because the web content could have changed considerably during the six months.

To experiment with content-based clusters, we have collected the page contents from the UACS web server in January 2003. Then the web logs of January 2003 was used as the training set, and the web logs of February 2003 (or the first week of February 2003) was used as the test set. To simplify the problem, we only retrieved contents of pages that have occurred in the training set. Using contents of pages of the entire web server (not just in the training set) may be beneficial because it gives us the ability to recommend pages that have not been accessed in the training set or even never been accessed before. However, this will also bring a much larger page set for analysis, which may cause the generated content-based clusters bigger in size and lower in quality.

In the following experiments, all the varieties of NCMs were tested and compared using the web logs of January 2003 as the training set and the web logs of February 2003 as the test set. In the test set, we also restricted maximum session length to 100. And we used only those sessions with a minimum session length of three, because only such sessions have the potential for navigation improvement. Moreover, those purely usage-data-based NCMs were also tested on the previous log data (from January 2001 to December 2002).

7.3.2 Experiment Plan

Our experiments involve enormous parameter combinations regarding different phases in the web mining process, including content page determination, learning, navigation compression, and evaluation, as shown in Figure 7.2. These parameters are described in detail in the following sections.

7.3.2.1 Content Page Determination

We have previously presented three different methods for content page identification: Reference Length (RL), Maximal Forward Reference (MFR), and MFR-RL. In the following experiments those content pages identified from different methods are further refined as follows:

- Each content page should meet a minimum support. As defined in Section 5.3.2, the support of a content page is the number of times this page has been visited as a content page. This restriction is applied to eliminate those content pages that might be caused by users' occasional activities.

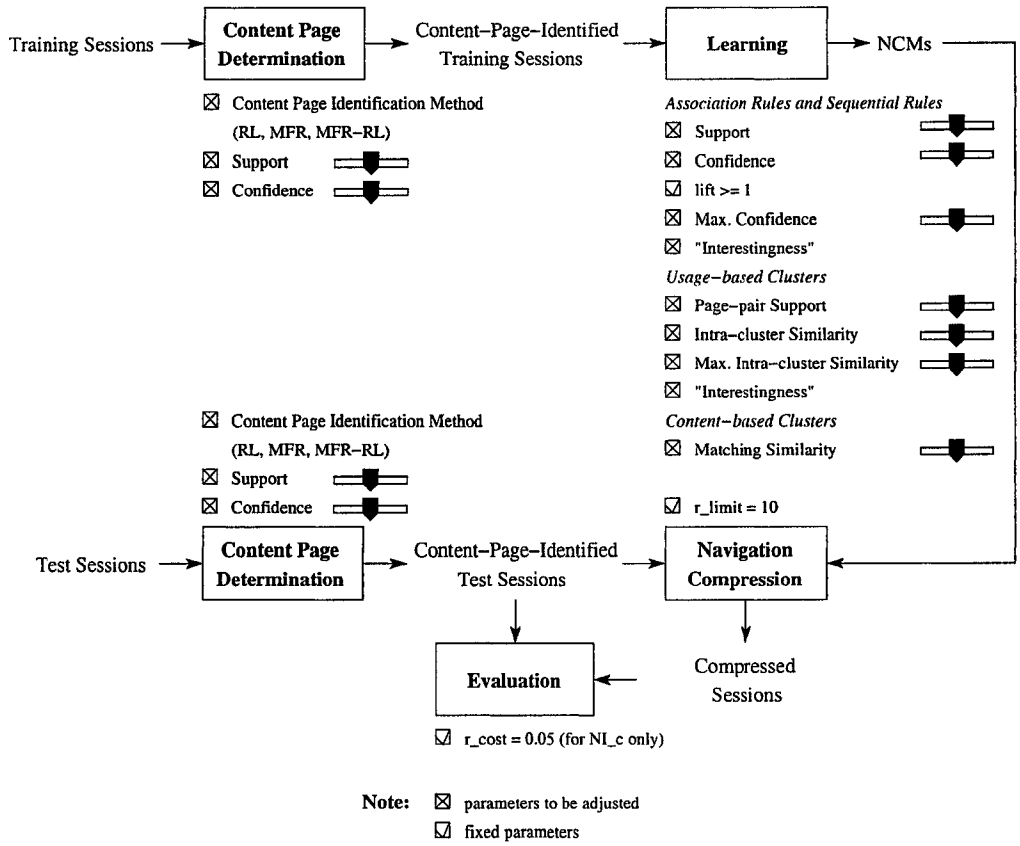


Figure 7.2: Experiment Plan and Parameter Control

Table 7.1: Number of Content Pages obtained with RL, MFR, and MFR-RL

support = 2, confidence = 5%

Content Page Set	***Number of Content Pages***		
	RL	MFR	MFR-RL
Recommendation Content pages (from training set)	6369	6152	*3642*
Test Content Pages (from test set)	5983	5834	*3238*

- Each content page should also meet a minimum confidence. As defined in Section 5.3.2, the confidence of a content page is the probability that this page was visited as a content page, and is calculated as

$$\frac{\text{number of times this page has been visited as a content page}}{\text{total number of times this page has been visited}}$$

  This restriction is applied to maintain the quality of the identified content pages.
- The number of content pages obtained with various identification methods (RL, MFR, and MFR-RL) could be different. Such being the case, to compare these methods we simply restrict their content page number to an arbitrary value N, e.g., the minimum among all those numbers. If the original content page number of a specific method is larger than N, then its N most significant content pages (based on their significance values) will be selected. In this way the comparison between different content page identification methods is accomplished by comparing the experimental results of their most important content pages.

In our experiments with a fixed number of content pages, we restricted the minimum support of content pages to two, and the minimum confidence to 5%. The number of recommendation content pages and test content pages obtained with RL, MFR, and MFR-RL are shown in Table 7.1. As previously discussed, the content page number of RL and MFR are then restricted to the number of corresponding MFR-RL-based content pages, by selecting their most significant content pages.

7.3.2.2 Learning

In the learning process, we can generate different numbers of association rules or sequential rules by adjusting the minimum support and confidence thresholds. To simplify the problem, the minimum lift of rules is set to a fixed value: $lift \geq 1$. Moreover, the maximum confidence threshold gives us the capability to exclude those rules with very high confidence, since we believe such rules may have very limited use for the purpose of navigation improvement. Also we will explore the effect of "interesting" rules on navigation improvement, e.g., if we can maintain a similar navigation improvement by excluding all the uninteresting rules and keeping only those interesting rules.

To generate different numbers of usage-based page clusters, we can adjust the minimum page-pair support and the minimum intra-cluster similarity. And the maximum intra-cluster similarity threshold lets us control the maximum similarity between cluster members. Moreover, the effect of "interesting" clusters will also be explored.

To generate content-based page clusters, the CBC algorithm itself involves an enormous number of parameters. To simplify the problem, in our experiments we use a fixed set of content-based clusters generated using CBC algorithm with a fixed setting (its default setting) of parameters. Given this fixed set of content-based clusters and any set of usage-based clusters, we can generate different sets of content-cluster-based NCMs by adjusting the minimum matching similarity between content-based and usage-based clusters.

7.3.2.3 Navigation Compression

The only parameter involved in this process is the maximum recommendation number, which is set to a fixed value: $r\_limit = 10$.

7.3.2.4 Evaluation

An important parameter in the evaluation phase is the recommendation cost used for calculating NI_c. In our experiments this recommendation cost is set to a fixed value for simplicity: $r\_cost = 0.05$. However, we do discuss how this parameter will affect the results of NI_c, and accordingly affect the comparison of different kinds of NCMs on their potentials in NI_c.

Furthermore, to capture the navigation improvements in different circumstances, each NI (or NI_c) is represented as three values $[v/v_c/v_r]$ (we expect that $v \leq v_c \leq v_r$):

- v — navigation improvement on all sessions.
- v_c — navigation improvement on those sessions with at least one content page.
- v_r — navigation improvement on those sessions with at least one correct recommendation.

Moreover, the precision and recall of the recommendations are computed as follows:

$$precision = \frac{\text{number of correct recommendations}}{\text{total number of recommendations}}$$

$$recall = \frac{\text{number of correct recommendations}}{\text{total number of content page occurrences}}$$

7.4 Experimental Results

7.4.1 Association Rules

We have previously presented three kinds of association rules that can be used as NCMs: content-page-insensitive, content-page-oriented, and content-page-only (as defined in Section 6.1.6). And these association rules can be further refined by selecting only those interesting rules ($interestingness = 1$). Moreover, the number of

association rules can be adjusted by changing the support and confidence thresholds ($lift \geq 1$).

Since the recommendations are restricted to content pages only, and the content page set used for recommendations has been pre-determined, in the following experiments we generate only those association rules that have only such pre-determined content pages as consequents.

7.4.1.1 Experiment 1: Support and Confidence Settings

Using the web logs of January 2003 as the training set and the web logs of February 2003 as the test set, we first experimented with association rules generated from different support and confidence settings. While the confidence parameter was adjusted between 5% and 75%, to compare the results the support parameter was set accordingly to assure that there were approximately 3000 rules generated. We could not guarantee generating exactly 3000 rules with a specific support and confidence setting because some rules may have exactly the same support. The results of different kinds of association rules (content-page-insensitive, content-page-oriented, and content-page-only) are shown respectively in Table 7.2, Table 7.3, and Table 7.4.

Summary of Experiment 1

(1) In spite of the type of association rules used (content-page-insensitive, content-page-oriented, and content-page-only) and the settings of support and confidence, the overall navigation improvement we can expect from the UACS data set is fairly small. The NI over those content-page-visited sessions $[NI(v_c)]$ is less than 5%, and the NI over all test sessions $[NI(v)]$ is less than 3%.

There are several possible reasons for this. First, the coverage of association rules is generally small. For example, a 3-itemset can generate up to 12 association rules (9 after rule reduction) but only have a coverage of three. And the consequent coverage can be even smaller. Therefore, it is typical for a large number of association rules to have a fairly small coverage. For example, the coverage of 3000 rules is less than 250, and the corresponding consequent coverage is less than 200. Consequently, the number of content pages that can be recommended is highly limited.

Another possible reason might be that users' interest in the UACS website is very diverse, and the content pages we have heuristically identified only represent a very small part of it. For this reason, the number of sessions where our recommendations do make sense is usually very small. Therefore, if we count only those sessions with at least one correct recommendation, the computed navigation improvement $[NI(v_r)]$ is much higher.

Also, user's interest might change over time. Therefore, a content page in the training set may not be a content page anymore in the test set. We found that only about 50% of the content pages in the training set would also appear as content pages in the test set, and less than 60% of the content pages in the test set can be found also as content pages in the training set.

It is also possible that a significant number of users (e.g., UACS faculty, staff, graduate and undergraduate students) are very familiar with the web resources in the UACS website. Therefore, their web navigation activities do not need recommendations, or have very limited potential for improvement with the help of recommendations.

Table 7.2: Experimental Results for Association-Rule-based NCMs (AR-1.1)

content-page-insensitive association rules

RL

Sup.	***Conf.***	#Rules	Coverage (total / consequent)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
0.1550%	*5%*	2980	238 / 182	2.60% 4.71% 15.78%	0.50% 2.93% 12.03%	6.29% $(\frac{9814}{156136})$	23.85% $(\frac{9814}{41148})$
0.1540%	*25%*	3013	241 / 160	2.17% 3.92% 15.36%	1.07% 2.89% 12.87%	9.91% $(\frac{8060}{81360})$	19.59% $(\frac{8060}{41148})$
0.1520%	*50%*	3017	222 / 115	1.48% 2.68% 14.87%	0.99% 2.18% 13.26%	14.63% $(\frac{5311}{36298})$	12.91% $(\frac{5311}{41148})$
0.1489%	*75%*	3007	190 / 76	0.89% 1.61% 17.66%	0.69% 1.39% 16.27%	17.71% $(\frac{2606}{14717})$	6.33% $(\frac{2606}{41148})$

MFR

Sup.	***Conf.***	#Rules	Coverage (total / consequent)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
0.1520%	*5%*	3002	222 / 153	2.27% 4.38% 15.09%	0.56% 2.82% 11.85%	7.95% $(\frac{10073}{126681})$	26.79% $(\frac{10073}{37598})$
0.1510%	*25%*	3070	200 / 127	1.90% 3.65% 14.95%	1.09% 2.76% 12.52%	14.16% $(\frac{8419}{59472})$	22.39% $(\frac{8419}{37598})$
0.1490%	*50%*	3070	154 / 81	1.21% 2.33% 13.89%	0.83% 1.86% 12.19%	19.71% $(\frac{5503}{27914})$	14.64% $(\frac{5503}{37598})$
0.1460%	*75%*	3330	114 / 54	0.73% 1.40% 19.23%	0.57% 1.19% 17.47%	23.50% $(\frac{2695}{11468})$	7.17% $(\frac{2695}{37598})$

MFR-RL

Sup.	***Conf.***	#Rules	Coverage (total / consequent)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
0.1500%	*5%*	3119	198 / 122	1.86% 4.24% 17.43%	0.64% 3.13% 14.63%	6.72% $(\frac{6087}{90586})$	25.42% $(\frac{6087}{23945})$
0.1490%	*25%*	3182	178 / 101	1.57% 3.58% 16.96%	0.88% 2.85% 14.55%	10.24% $(\frac{5237}{51123})$	21.87% $(\frac{5237}{23945})$
0.1489%	*50%*	3112	129 / 62	0.92% 2.11% 15.26%	0.60% 1.70% 13.43%	13.91% $(\frac{3320}{23867})$	13.87% $(\frac{3320}{23945})$
0.1450%	*75%*	3441	82 / 39	0.47% 1.08% 17.38%	0.34% 0.88% 15.48%	16.55% $(\frac{1567}{9470})$	6.54% $(\frac{1567}{23945})$

Table 7.3: Experimental Results for Association-Rule-based NCMs (AR-1.2)

content-page-oriented association rules

RL

Sup.	***Conf.***	#Rules	Coverage (total / consequent)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
0.038%	*5%*	3230	232 / 135	2.29% 4.15% 15.89%	1.41% 3.25% 13.55%	12.84% $(\frac{8398}{65382})$	20.41% $(\frac{8398}{41148})$
0.037%	*25%*	3043	116 / 58	1.09% 1.98% 18.29%	0.89% 1.66% 16.28%	24.33% $(\frac{3728}{15321})$	9.06% $(\frac{3728}{41148})$
0.033%	*50%*	3266	54 / 30	0.44% 0.79% 18.68%	0.37% 0.68% 16.95%	29.55% $(\frac{1462}{4947})$	3.55% $(\frac{1462}{41148})$
0.030%	*75%*	3126	27 / 16	0.15% 0.26% 13.81%	0.13% 0.23% 12.66%	41.25% $(\frac{526}{1275})$	1.28% $(\frac{526}{41148})$

MFR

Sup.	***Conf.***	#Rules	Coverage (total / consequent)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
0.073%	*5%*	3247	101 / 65	1.69% 3.25% 15.55%	1.17% 2.59% 13.16%	20.27% $(\frac{7766}{38318})$	20.66% $(\frac{7766}{37598})$
0.073%	*25%*	3135	44 / 26	1.10% 2.12% 22.14%	0.92% 1.79% 19.56%	30.59% $(\frac{4184}{13676})$	11.13% $(\frac{4184}{37598})$
0.072%	*50%*	3222	34 / 21	0.90% 1.73% 24.03%	0.80% 1.54% 22.08%	37.85% $(\frac{2892}{7641})$	7.69% $(\frac{2892}{37598})$
0.070%	*75%*	3239	24 / 13	0.42% 0.81% 22.33%	0.38% 0.74% 20.75%	51.53% $(\frac{1449}{2812})$	3.85% $(\frac{1449}{37598})$

MFR-RL

Sup.	***Conf.***	#Rules	Coverage (total / consequent)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
0.025%	*5%*	3443	233 / 122	1.83% 4.18% 17.49%	1.14% 3.37% 15.11%	11.66% $(\frac{6031}{51713})$	25.19% $(\frac{6031}{23945})$
0.024%	*25%*	3167	98 / 48	0.84% 1.92% 20.72%	0.69% 1.63% 18.95%	18.92% $(\frac{2109}{11149})$	8.81% $(\frac{2109}{23945})$
0.022%	*50%*	3334	47 / 28	0.41% 0.94% 18.90%	0.36% 0.82% 17.45%	26.61% $(\frac{1058}{3976})$	4.42% $(\frac{1058}{23945})$
0.020%	*75%*	3443	37 / 15	0.14% 0.31% 16.25%	0.13% 0.29% 15.22%	46.59% $(\frac{430}{923})$	1.80% $(\frac{430}{23945})$

Table 7.4: Experimental Results for Association-Rule-based NCMs (AR-1.3)

content-page-only association rules

RL

Sup.	***Conf.***	#Rules	Coverage (total / consequent)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
0.024%	*5%*	3279	75 / 73	0.94% 1.70% 21.15%	0.63% 1.14% 18.11%	13.23% $(\frac{3079}{23265})$	7.48% $(\frac{3079}{41148})$
0.024%	*25%*	3212	66 / 48	0.80% 1.44% 21.96%	0.61% 1.10% 19.17%	18.71% $(\frac{2591}{13849})$	6.30% $(\frac{2591}{41148})$
0.023%	*50%*	3159	41 / 31	0.44% 0.80% 18.21%	0.36% 0.66% 16.17%	26.80% $(\frac{1577}{5885})$	3.83% $(\frac{1577}{41148})$
0.020%	*75%*	3287	33 / 21	0.16% 0.29% 13.21%	0.13% 0.24% 11.79%	32.09% $(\frac{638}{1988})$	1.55% $(\frac{638}{41148})$

MFR

Sup.	***Conf.***	#Rules	Coverage (total / consequent)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
0.048%	*5%*	3202	33 / 33	1.15% 2.22% 27.25%	0.94% 1.81% 24.21%	22.56% $(\frac{3576}{15853})$	9.51% $(\frac{3576}{37598})$
0.048%	*25%*	3197	33 / 33	1.14% 2.19% 27.07%	0.93% 1.80% 24.10%	23.21% $(\frac{3539}{15245})$	9.41% $(\frac{3539}{37598})$
0.047%	*50%*	3174	32 / 23	0.87% 1.67% 23.56%	0.75% 1.45% 21.35%	33.60% $(\frac{2858}{8505})$	7.60% $(\frac{2858}{37598})$
0.042%	*75%*	3233	23 / 14	0.45% 0.86% 20.51%	0.40% 0.77% 18.93%	45.12% $(\frac{1507}{3340})$	4.01% $(\frac{1507}{37598})$

MFR-RL

Sup.	***Conf.***	#Rules	Coverage (total / consequent)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
0.013%	*5%*	3286	69 / 67	0.96% 2.18% 26.07%	0.75% 1.72% 23.34%	14.32% $(\frac{2154}{15040})$	9.00% $(\frac{2154}{23945})$
0.013%	*25%*	3213	63 / 56	0.79% 1.80% 24.68%	0.64% 1.47% 22.16%	17.59% $(\frac{1906}{10835})$	7.96% $(\frac{1906}{23945})$
0.012%	*50%*	3287	47 / 37	0.39% 0.90% 18.81%	0.33% 0.76% 17.06%	23.31% $(\frac{1028}{4411})$	4.29% $(\frac{1028}{23945})$
0.008%	*75%*	3179	48 / 29	0.16% 0.36% 14.48%	0.14% 0.31% 13.37%	29.73% $(\frac{459}{1544})$	1.92% $(\frac{459}{23945})$

(2) There are a significant proportion of sessions (more than 50% of all the test sessions) in which the users did not visit any content page. These sessions dramatically reduced the overall navigation improvement because there would be no "correct" recommendation on these sessions. This is shown by the observation that $NI(v_c)$ is much higher than the corresponding $NI(v)$. If we assume that only users visiting content pages are interested in using the recommendations, then $NI(v_c)$ is probably a better measure for the "true" improvement than $NI(v)$.

(3) Taking recommendation cost into account can lead to completely different evaluation results. For example, using content-page-insensitive rules and RL-based content pages (as shown in Table 7.2), the overall NI $[NI(v)]$ obtained with $support = 0.1550\%$ and $confidence = 5\%$ is higher than the overall NI obtained with $support = 0.1540\%$ and $confidence = 25\%$. Note that the result is totally different when taking recommendation cost into account: the corresponding NI_c of the latter case is much higher than the corresponding NI_c of the former case. This is because the former case achieves a slightly better NI (with approximately 2000 more correct recommendations) at a much higher cost (about 75000 more total recommendations).

(4) Precision and recall can not be used in isolation to evaluate the actual navigation improvement measured by the number of traversal links. For example, using content-page-insensitive rules with confidence set to 50% (as shown in Table 7.2), the precision and recall obtained with MFR-based content pages are both higher than those obtained with RL-based content pages. However, the navigation improvements (both NI and NI_c) obtained with RL are higher than those obtained with MFR.

(5) Association rules generated with higher support and lower confidence tend to have better results in navigation improvement. This is probably because the association rule set generated with higher support and lower confidence tends to have a larger coverage, therefore can potentially make recommendations on more content pages. However, the reason why there exists such a relationship between support-confidence setting and rule coverage is still unclear.

(6) Association rules generated with lower support and higher confidence tend to result in higher precision but lower recall.

(7) Association rules with very high confidence values are generally less useful. For example, we found many occurrences of the following patterns: suppose most visits to page b came from page a, then we can obtain two rules $R_1 = \{a \rightarrow b\}(confidence = c_1)$ and $R_2 = \{b \rightarrow a\}(confidence = c_2)$, where c_2 could be much higher than c_1, e.g., higher than 95%. However, it is apparent that R_2 is not as useful as R_1 because page a is visited before page b most of the time. Here, we call those association rules like R_2 *backtracking rules*. We expect that this problem can be addressed by sequential rules.

(8) The navigation improvements obtained with different kinds of association rules are also quite different. Content-page-insensitive rules tend to have the best results in NI, and content-page-oriented rules tend to have the best results in NI_c. And content-page-only rules tend to have the worst overall performance. Moreover, while content-page-oriented and content-page-only rules tend to result in higher precision than content-page-insensitive rules, at the same time they tend to result in lower recall.

Content-page-oriented and content-page-only rules are much more cautious about generating recommendations than content-page-insensitive rules. For example, with approximately 3000 association rules generated with RL-based content pages and $confidence = 25\%$, content-page-insensitive rules generated 81360 recommendations, while content-page-oriented and content-page-only rules only generated 15321 and 13849 recommendations, respectively. Therefore, we can expect that a different r_cost setting will being a much larger impact on the NI_c results of content-page-insensitive rules than on the other two.

The navigation improvement obtained with content-page-insensitive rules is less sensitive to support and confidence settings than those obtained with content-page-oriented and content-page-only rules. In other words, the navigation improvement obtained with content-page-insensitive rules tend to change less with different support and confidence settings, compared to those obtained with content-page-oriented and content-page-only rules.

(9) Among the three content page identification methods (RL, MFR, and MFR-RL), RL tends to have the best performance (i.e., potential for best navigation improvement) with association-rule-based NCMs. However, the navigation improvement obtained with RL is not considerably higher than those obtained with the other two methods (MFR and MFR-RL).

In addition, the navigation improvement obtained with MFR is less sensitive to the association rule's support and confidence settings no matter which kind of rules are used, while those obtained with RL and MFR-RL are much more sensitive with content-page-oriented and content-page-only rules.

In the following experiments we tested other features of association-rule-based NCMs with content-page-insensitive rules and RL-based content pages.

7.4.1.2 Experiment 2: Maximum Confidence Threshold

In the previous experiment we have found that association rules with very high confidence values are generally less useful. Therefore, in this experiment we generated association rules limited to a maximum confidence threshold: all those association rules with a confidence higher than the threshold are not generated. While the maximum confidence threshold was adjusted between 100% and 50%, the minimum support and confidence were fixed: $support = 0.1540\%$, $confidence = 25\%$. The results of this experiment are shown in Table 7.5.

The results confirm that association rules with very high confidence values are less useful for navigation improvement. This is largely because the recommendations generated from these rules, though tend to have slightly higher precision, are very limited in volume. In other words, they are rarely used in helping users' navigation.

The results show that by excluding those association rules with confidence higher than 90%, we can reduce the number of rules by 50% and still obtain more than 98% of the improvement. Therefore, in the following experiments we tested other features of association-rule-based NCMs by restricting the maximum confidence to 90%.

7.4.1.3 Experiment 3: "Interesting" Association Rules

We have previously defined interestingness of association rules based on the hyperlink structure of the web site (see Section 6.1.1). And we expect those "interesting"

Table 7.5: Experimental Results for Maximum Confidence Threshold

content-page-insensitive association rules
(min_support = 0.1540% , min_confidence = 25%)

RL-based content pages

Max. Conf.	#Rules	Coverage (total / consequent)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
100%	3013	241 / 160	2.17% 3.92% 15.36%	1.07% 2.89% 12.87%	9.91% $(\frac{8060}{81360})$	19.59% $(\frac{8060}{41148})$
90%	1596	215 / 155	2.14% 3.87% 15.34%	1.05% 2.85% 12.83%	9.87% $(\frac{7988}{80913})$	19.41% $(\frac{7988}{41148})$
75%	864	194 / 141	1.94% 3.51% 15.29%	0.91% 2.53% 12.82%	9.44% $(\frac{7233}{76655})$	17.58% $(\frac{7233}{41148})$
50%	296	149 / 100	1.47% 2.65% 16.27%	0.64% 1.87% 13.96%	8.09% $(\frac{4952}{61208})$	12.03% $(\frac{4952}{41148})$

Table 7.6: Experimental Results for "Interesting" Association Rules

content-page-insensitive association rules
(min_support = 0.1540% , min_confidence = 25%, max_confidence = 90%)

RL-based content pages

Type of Rules	#Rules	Coverage (total / consequent)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
all rules	1596	215 / 155	2.14% 3.87% 15.34%	1.05% 2.85% 12.83%	9.87% $(\frac{7988}{80913})$	19.41% $(\frac{7988}{41148})$
"interesting" rules *(interestingness = 1)*	706	124 / 73	1.45% 2.62% 27.00%	1.01% 2.17% 25.03%	9.09% $(\frac{2969}{32677})$	7.22% $(\frac{2969}{41148})$

rules (measured by interestingness) to be more useful for navigation improvement than uninteresting rules. In this experiment we generated only interesting rules and compared the navigation improvement with that obtained with all the rules. The result of this experiment is shown in Table 7.6.

The result of this experiment shows that recommendations generated from "interesting" rules have the potential to gain higher navigation improvement than those recommendations generated from uninteresting rules, if only the correct recommendations have been made. This is demonstrated by the observation that the $NI(v_r)$ and $NI_c(v_r)$ of interesting rules are much higher (about 100% higher) than those obtained with all rules. Moreover, by using "interesting" rules only we can reduce the number of rules by more than 50%, reduce the number of recommendations by 60%, and still obtain 70% of the improvement.

Note that uninteresting rules are still useful because there is a remarkable portion (more than 30%) of the navigation improvement not covered by interesting rules. This implies that even if the relevant page is linked by the current page the user is

Table 7.7: Experimental Results for Varying Number of Association Rules

content-page-insensitive association rules
(min_confidence = 25%, max_confidence = 90%)

RL-based content pages

Support	#Rules	Coverage (total / consequent)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
0.199%	502	156 / 102	1.87% 3.38% 15.76%	0.98% 2.56% 13.37%	10.62% $(\frac{7027}{66193})$	17.08% $(\frac{7027}{41148})$
0.169%	1017	198 / 141	2.08% 3.76% 15.37%	1.04% 2.79% 12.91%	10.11% $(\frac{7783}{76994})$	18.91% $(\frac{7783}{41148})$
0.147%	2058	230 / 166	2.18% 3.93% 15.14%	1.04% 2.87% 12.62%	9.75% $(\frac{8194}{84015})$	19.91% $(\frac{8194}{41148})$
0.136%	3042	249 / 178	2.22% 4.02% 15.34%	1.04% 2.92% 12.77%	9.48% $(\frac{8291}{87417})$	20.15% $(\frac{8291}{41148})$
0.122%	5016	281 / 201	2.36% 4.27% 15.36%	1.13% 3.12% 12.83%	9.39% $(\frac{8608}{91694})$	20.92% $(\frac{8608}{41148})$
0.071%	10043	477 / 329	2.74% 4.95% 15.20%	1.23% 3.52% 12.51%	9.02% $(\frac{10076}{111758})$	24.49% $(\frac{10076}{41148})$

viewing, sometimes the user may miss these "correct" links. This is probably because many web pages are designed with a complex layout and a relatively large volume of content, which makes the relevant information hard to find.

7.4.1.4 Experiment 4: Varying Number of Association Rules

In this experiment we tested how navigation is improved with a varying number of association rules. Here we set the minimum confidence to 25%, set the maximum confidence to 90%, and adjust only the support. The results of this experiment are shown in Table 7.7.

The results show that with a given set of content pages and a varying number of association rules, $NI(v_r)$ and $NI_c(v_r)$ do not change much, while $NI(v)$, $NI(v_c)$ and $NI_c(v)$, $NI_c(v_c)$ are improved more or less when the number of rules becomes larger, as illustrated in Figure 7.3. It is worth noting that more rules generally lead to more recommendations. Therefore, the result of NI_c could be different with different settings of r_cost.

Note that not all rules have the same effect on navigation improvement: rules with higher support (e.g., $support \geq 0.199\%$) tend to be more useful, while rules with lower support (e.g., $support \leq 0.169\%$) tend to be less useful. Moreover, some rules (e.g., $0.136\% \leq support \leq 0.147\%$) have very limited effect on navigation improvement. This is probably because these rules mainly extend the rule set with the page coverage we have already had, and introduce very few new pages into the coverage.

The results also show that the precision becomes lower with more rules (lower

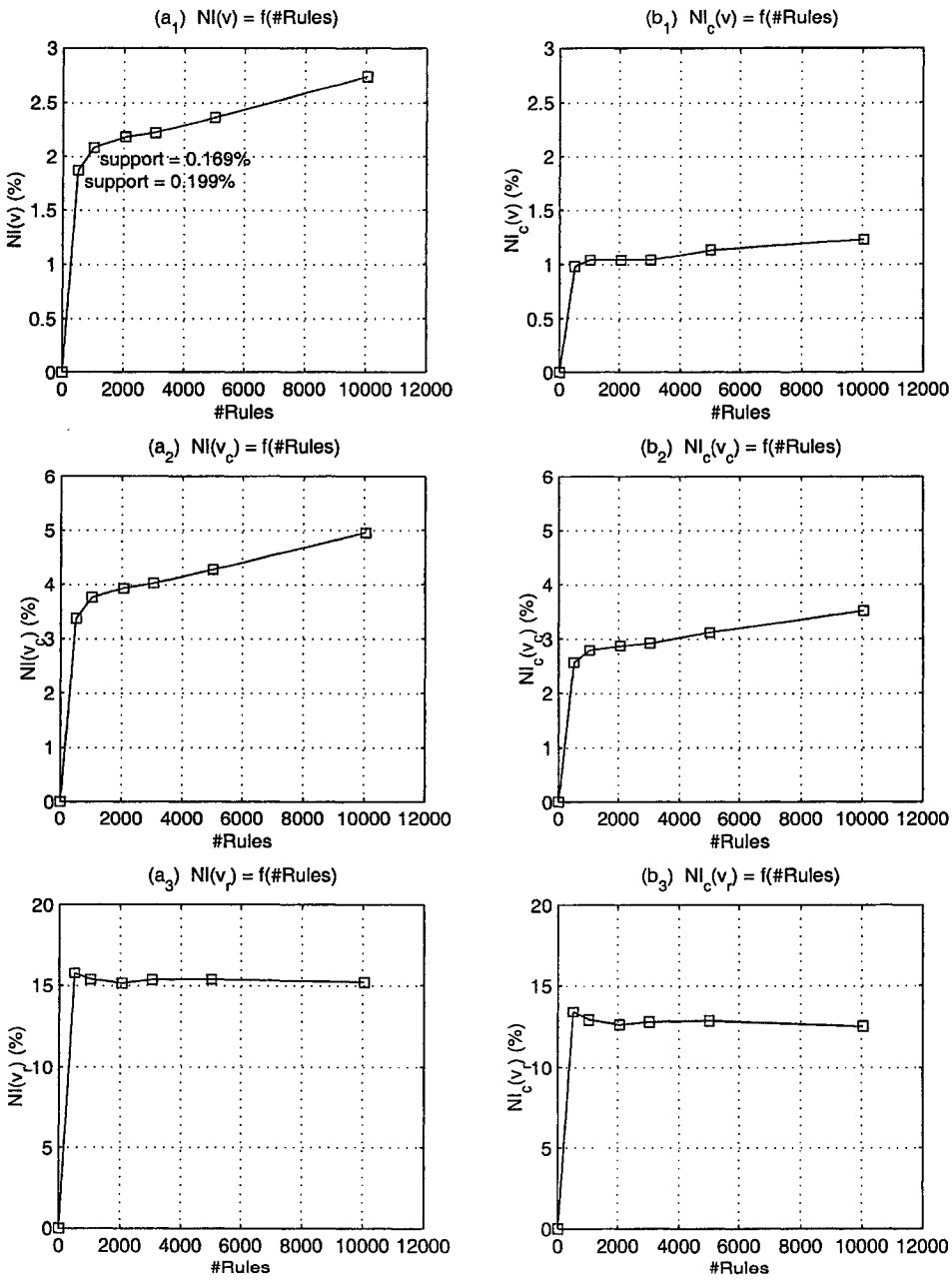


Figure 7.3: Navigation Improvement vs. #Association Rules

Table 7.8: Experimental Results for Varying Number of Content Pages

content-page-insensitive association rules
(min_ confidence = 25%, max_ confidence = 90%)

RL-based content pages
(support = 2)

Association Rule			Content Page		Evaluation			
Sup.	#Rules	Coverage	***Conf.***	#Content Pages (training / test)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
0.147%	2061	230 / 167	*5%*	6369 / 5983	2.17% 3.54% 15.01%	1.03% 2.56% 12.48%	9.75% $(\frac{8243}{84523})$	17.06% $(\frac{8243}{48304})$
0.145%	2011	200 / 106	*10%*	5499 / 5086	1.94% 3.48% 17.29%	1.21% 2.78% 14.91%	12.23% $(\frac{6627}{54205})$	16.20% $(\frac{6627}{40899})$
0.142%	2056	121 / 67	*15%*	4586 / 4136	1.27% 2.67% 17.54%	0.77% 2.10% 14.67%	13.53% $(\frac{5078}{37523})$	15.59% $(\frac{5078}{32564})$
0.137%	2049	74 / 42	*20%*	3576 / 3075	1.17% 2.90% 18.89%	0.82% 2.35% 15.96%	17.19% $(\frac{4559}{26519})$	17.87% $(\frac{4559}{25516})$
0.135%	2058	49 / 30	*25%*	2994 / 2532	1.04% 2.94% 19.86%	0.78% 2.43% 16.90%	19.74% $(\frac{3810}{19305})$	18.14% $(\frac{3810}{21001})$

support), while the recall becomes higher.

7.4.1.5 Experiment 5: Varying Number of Content Pages

So far we have used a fixed set of content pages for all the previous experiments of association-rule-based NCMs. In this experiment we tested how navigation is improved with a varying number of content pages.

A larger number of content pages can have two effects on navigation improvement: a positive one is that more paths can be compressed (no recommendation will be made if it is not for a content page, therefore no compression); on the other hand, there are also more pages in the paths we can not skip.

The content page set we used for this experiment was RL-based. Here we report two experiments. In the first experiment, we set the minimum support of content pages to two, and adjusted the number of content pages by changing the confidence threshold.[1] In the second experiment, we set the minimum confidence of content pages to 5%, and adjusted the number of content pages by changing the support threshold. For each content page set we generated approximately 2000 content-page-insensitive rules (with minimum confidence set to 25%, maximum confidence set to 90%, and minimum support adjusted accordingly).

The results of the first experiment are shown in Table 7.8. In this experiment, we found that the rule coverage became much smaller when the confidence of content pages was set to a higher value. This means that there exist many content pages with

[1]Note that the support and confidence of content pages (as defined in Section 5.3.2) are different with the support and confidence of association rules.

fairly low confidence but visited a lot (they need to be visited frequently enough to be included in the 2000 association rules). And the results of this experiment show that these content pages, although are visited much more often as auxiliary pages than as content pages, can still be very useful for navigation improvement.

In the second experiment, when we adjusted the support of content pages from 2 to 15, the obtained association rule set was unchanged. This is because the 2061 rules we generated have a fairly small coverage (230 pages), and all the content pages covered by these rules tend to have a relatively high support (>15). Such being the case, when we adjust the number of content pages with support from 2 to 15, having fewer content pages always means that there are more pages to skip without losing any opportunity of recommendation. Therefore, in this particular case navigation improvement is always better when the number of content pages becomes smaller.

We didn't experiment with support higher than 15 because in that case, the number of content pages would be too small (less than 900 content pages in more than 35000 total pages for both training and test set) so that the experiment itself would lose its practical meaning. The coverage of association rules can be increased by generating more rules. However, it takes a much larger number of rules to considerably increase the coverage, which will inevitably impair the performance of real-time recommendation. This again clearly demonstrates the limitation of association-rule-based NCMs.

In this experiment we have found that: (1) content pages with fairly low confidence are still useful for navigation improvement; and (2) for NCMs with relatively small coverage, setting the support of content pages higher (within a limit to maintain the practical meaning) will only make the result better. Therefore, in the experiments of other NCMs (where the coverage might be larger than that of association-rule-based NCMs, but is still relatively small) we used a fixed setting for content pages: $support = 2$ and $confidence = 5\%$.

7.4.1.6 Experiment 6: Time Effect

So far all the experiments we have conducted used one-month log data for training and one-month log data for testing. In other words, these experiments were based on the assumption that the knowledge of user navigation patterns is updated on a monthly basis. However, since users' interests might change over time, we can expect that the "interestingness" of the static knowledge will change over time as well. Therefore, it is possible that updating the knowledge on a more frequent basis will increase the navigation improvement that can be achieved.

In this experiment we tested the navigation improvement with association-rule-based NCMs updated on a weekly basis. The January web log was still used as training data, but only the first week of the February web log was used for testing. The result of this experiment is shown in Table 7.9.

We found that 80% of the content pages in the test set can also be found as content pages in the training set, while this rate for the monthly test set was less than 60%. However, the experimental result shows that the navigation improvement obtained from the weekly test set is similar to that obtained from the monthly test set. This implies that updating the knowledge more frequently will not necessarily increase the navigation improvement that can be achieved. Therefore, in the experiments of other NCMs we used only monthly log data.

Table 7.9: Experimental Results for Weekly Updated Knowledge

content-page-insensitive association rules (#Rule = 2061)
(support = 0.147%, min_ confidence = 25%, max_ confidence = 90%)

RL-based content pages
(support = 2, confidence = 5%)

Knowledge Update Frequency	#Content Pages (training / test)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
Monthly	6369 / 5983	2.17% 3.54% 15.01%	1.03% 2.56% 12.48%	9.75% $(\frac{8243}{84523})$	17.06% $(\frac{8243}{48304})$
Weekly	6369 / 2150	2.02% 3.66% 14.14%	0.86% 2.61% 11.71%	9.01% $(\frac{2013}{22336})$	18.67% $(\frac{2013}{10784})$

7.4.1.7 Experiment 7: More Training and Test Data

In this experiment we tested the navigation improvement of applying association-rule-based NCMs on more log data. The data we used was UACS log data from January 2001 to February 2003. And we assumed that the knowledge was updated on a monthly basis, i.e., in each case the previous one-month logs were used as training set, and the later one-month logs were used as test set. Moreover, we assumed that our recommendation system could handle 2000 association rules for real-time response. Therefore, in each case we only generated approximately 2000 association rules. The results of this experiment are shown in Table 7.10 and Table 7.11.

The results show that the navigation improvements obtained with association-rule-based NCMs from different monthly data sets are fairly consistent. With approximately 2000 association rules, the navigation improvement that can be obtained from the UACS log data is very limited: generally, $NI(v)$ is less than 2%, and $NI(v_c)$ is less than 3%. Moreover, the precision and recall of recommendations are also not high: both precision and recall are generally less than 15%.

7.4.1.8 Summary

- Association rules with higher support tend to be more useful for navigation improvement, while rules with very high confidence (e.g., higher than 90%) are generally less useful.
- Content-page-insensitive rules tend to have the best results in NI, and are generally less sensitive to different support and confidence settings. However, content-page-oriented and content-page-only rules are much more cautious about generating recommendations. Therefore, a different setting of r_cost will have a much larger impact on content-page-insensitive rules than on the other two.
- Given a set of association rules, its coverage is important in determining the navigation improvement that can be achieved. However, the coverage of association rules is generally very limited. And this problem can not be simply solved by generating a very large number of rules, because too many rules will

Table 7.10: Experimental Results on More UACS Log Data

content-page-insensitive association rules
(min_confidence = 25%, max_confidence = 90%)
(support adjusted to generate approximately 2000 association rules)

RL-based content pages
(support = 2, confidence = 5%)

Training Data	Test Data	#Rules	#Content Pages (training / test)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
01/2001	02/2001	2017	7938 / 7591	2.05% 2.95% 9.35%	0.36% 1.56% 6.95%	8.39% $(\frac{9766}{116444})$	15.77% $(\frac{9766}{61921})$
02/2001	03/2001	2023	7591 / 8554	1.74% 2.58% 8.75%	0.21% 1.36% 6.51%	8.80% $(\frac{9940}{112892})$	14.60% $(\frac{9940}{68089})$
03/2001	04/2001	2041	8554 / 8675	1.52% 2.07% 8.18%	0.30% 1.08% 6.13%	9.69% $(\frac{8614}{88929})$	10.53% $(\frac{8614}{81838})$
04/2001	05/2001	2004	8675 / 8906	1.50% 2.00% 7.95%	0.46% 1.06% 5.91%	11.71% $(\frac{7556}{64536})$	11.30% $(\frac{7556}{66890})$
05/2001	06/2001	2024	8906 / 7855	1.70% 2.24% 8.24%	0.57% 1.16% 5.91%	13.01% $(\frac{6591}{50674})$	13.01% $(\frac{6591}{50646})$
06/2001	07/2001	2036	7855 / 8085	1.54% 1.96% 7.51%	0.54% 0.98% 5.20%	15.54% $(\frac{7339}{47222})$	12.73% $(\frac{7339}{57633})$
07/2001	08/2001	2012	8085 / 9368	1.74% 2.22% 7.32%	0.56% 1.09% 5.12%	14.10% $(\frac{9494}{67337})$	13.55% $(\frac{9494}{70066})$
08/2001	09/2001	2036	9368 / 9096	1.52% 2.12% 8.63%	-0.02% 0.81% 5.88%	8.06% $(\frac{11322}{140396})$	13.43% $(\frac{11322}{84281})$
09/2001	10/2001	2001	9096 / 10291	1.49% 2.18% 9.39%	-0.12% 0.97% 6.95%	6.38% $(\frac{11375}{178394})$	12.19% $(\frac{11375}{93329})$
10/2001	11/2001	2001	10291 / 10280	1.45% 2.10% 8.78%	0.11% 0.95% 6.50%	8.59% $(\frac{10694}{124439})$	12.83% $(\frac{10694}{83352})$
11/2001	12/2001	2042	10280 / 9550	1.58% 2.22% 8.35%	0.38% 1.16% 6.34%	9.69% $(\frac{8939}{92235})$	12.92% $(\frac{8939}{69184})$
12/2001	01/2002	2024	9550 / 10616	1.30% 1.84% 8.25%	0.26% 0.87% 6.05%	9.35% $(\frac{9819}{105039})$	11.00% $(\frac{9819}{89244})$

Table 7.11: Experimental Results on More UACS Log Data (Continued)

content-page-insensitive association rules
(min_confidence = 25%, max_confidence = 90%)
(support adjusted to generate approximately 2000 association rules)

RL-based content pages
(support = 2, confidence = 5%)

Training Data	Test Data	#Rules	#Content Pages (training / test)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
01/2002	02/2002	2020	10616 / 9730	1.37% 2.33% 8.90%	0.19% 1.20% 6.60%	8.60% $(\frac{9815}{114109})$	13.76% $(\frac{9815}{71342})$
02/2002	03/2002	2008	9730 / 10440	1.74% 2.52% 9.88%	0.58% 1.50% 7.65%	10.49% $(\frac{11185}{106595})$	13.31% $(\frac{11185}{84043})$
03/2002	04/2002	2047	10440 / 11059	1.38% 1.98% 9.20%	0.46% 1.15% 7.12%	10.82% $(\frac{8834}{81609})$	10.22% $(\frac{8834}{86471})$
04/2002	05/2002	2011	11059 / 10368	1.83% 2.45% 9.72%	0.80% 1.46% 7.47%	14.11% $(\frac{10552}{74784})$	12.51% $(\frac{10552}{84370})$
05/2002	06/2002	2055	10368 / 9813	1.82% 2.75% 11.84%	0.98% 1.83% 8.96%	16.50% $(\frac{9389}{56895})$	13.02% $(\frac{9389}{72094})$
06/2002	07/2002	2027	9813 / 9420	1.52% 2.06% 13.98%	1.02% 1.55% 11.46%	20.09% $(\frac{6903}{34363})$	9.39% $(\frac{6903}{73542})$
07/2002	08/2002	2052	9420 / 6333	1.50% 2.30% 17.34%	0.97% 1.70% 14.12%	16.76% $(\frac{5743}{34265})$	11.55% $(\frac{5743}{49729})$
08/2002	09/2002	2010	6333 / 6783	1.04% 1.69% 19.71%	0.63% 1.24% 16.17%	14.17% $(\frac{5731}{40434})$	8.51% $(\frac{5731}{67325})$
09/2002	10/2002	2048	6783 / 7281	1.45% 2.30% 14.52%	0.73% 1.63% 11.76%	12.78% $(\frac{8827}{69048})$	13.45% $(\frac{8827}{65610})$
10/2002	11/2002	2041	7281 / 6953	1.43% 2.27% 14.40%	0.87% 1.71% 11.98%	14.97% $(\frac{7021}{46906})$	12.39% $(\frac{7021}{56682})$
11/2002	12/2002	2017	6953 / 6812	1.02% 1.70% 12.84%	0.62% 1.27% 10.80%	15.45% $(\frac{4856}{31424})$	9.31% $(\frac{4856}{52139})$
12/2002	01/2003	2034	6812 / 6369	1.38% 2.17% 13.95%	0.68% 1.51% 11.77%	10.48% $(\frac{6515}{62194})$	10.59% $(\frac{6515}{61536})$
01/2003	02/2003	2061	6369 / 5983	2.17% 3.54% 15.01%	1.03% 2.56% 12.48%	9.75% $(\frac{8243}{84523})$	17.06% $(\frac{8243}{48304})$

impact the performance of real-time recommendation.

- Users often miss those "correct" links which can lead them to the pages of interest. Therefore, recommending a page which is already linked by the current page the user is viewing can still be useful.
- Finally, our experimental results are indeed determined by a number of heuristics and parameter settings, including the heuristics for user identification, session identification, content page identification, and the settings of r_cost and r_limit. Different choices on these heuristics and parameter settings can undoubtedly lead to different results. However, the appropriate choices of these heuristics and parameters should be determined by the status of the website and user preferences, and so far can only be obtained empirically.

7.4.2 Sequential Rules

Our sequential rules are generated with the same basic algorithm (DHP) previously used for generating association rules, only with additional sequential information. Therefore, we can expect sequential-rule-based NCMs to have a number of common features with association-rule-based NCMs.

Our experiments showed that sequential-rule-based NCMs do have very similar performance on navigation improvement with association-rule-based NCMs. The overall navigation improvements $[NI(v)]$ obtained with sequential-rule-based NCMs and association-rule-based NCMs are quite similar. The only difference is that content-page-only sequential rules perform much better than content-page-only association rules. However, the reason why there is such a difference is still unclear.

Besides, sequential-rule-based NCMs tend to generate recommendations more cautiously and with higher precision. This is probably because *backtracking rules* are not generated with sequential rules. Consequently, sequential-rule-based NCMs generally have higher $NI_c(v)$ than association-rule-based NCMs.

7.4.3 Combined Rules

When multiple rules (association rules or sequential rules) have the same antecedent but different consequents, we can combine these rules into one to improve the performance of real-time recommendation.

As previously discussed, we generate dynamic recommendations by applying NCMs to the user's current path. For a rule-based NCM, we match its antecedent with the user's current path, and recommend those pages in its consequent if the matching is approved. Therefore, by combining those rules with the same antecedent we can assure that each distinct antecedent is matched only once, instead of being matched multiple times as with the traditional rule-based NCMs.

A combined rule is represented as:

$$a_1, \cdots, a_m \rightarrow b_1(\lambda_1), \cdots b_n(\lambda_n)$$

where each b_i in the consequent is associated with a value λ_i, which is the significance of the corresponding rule $\{a_1, \cdots, a_m \rightarrow b_i\}$.

Combined rules are a special kind of rules created for our recommendation mechanism. They do not have those criteria such as support, confidence, and lift in traditional rules, but keep all the necessary information (antecedent, consequent, and

significance) to be used as NCMs. Moreover, unlike traditional rules, those items appearing in the consequent of the same combined rule do not necessarily carry co-occurrence relationship between each other.

Our experiments showed that by combining rules, we can further reduce the number of association rules (after rule reduction) by 50~65%, and reduce the number of sequential rules (also after rule reduction) by 40~55%.

A combined rule can be considered as a combined form of multiple rules, or as a single rule by itself when used as a NCM. Each of these two interpretations has its own cognitive concerns. In a sense combined rules have improved the low-coverage problem of traditional rules. Even so, the coverage of combined rules is still much smaller than that of clusters, which will be explored in the following sections.

In our experiments we consider combined rules only as a way of improving the recommendation performance of rule-based NCMs, and still use the features of traditional rules when comparing rule-based NCMs with other kinds of NCMs.

7.4.4 Usage-based Clusters

We have previously presented three kinds of usage-based page clusters that can be used as NCMs: content-page-insensitive, content-page-oriented, and content-page-only (as defined in Section 6.3.1.5). And the number of clusters can be adjusted by changing the similarity threshold (the minimum similarity between two members of the same cluster) and the minimum support of page pairs.

Since the recommendations are restricted to content pages only, and the content page set used for recommendations has been pre-determined, in the following experiments we generate only those clusters that contain such content pages. To compare the results of different content page identification methods (RL, MFR, and MFR-RL), we still used the content page sets shown in Table 7.1.

We first experimented with page clusters obtained with the generalized PageGather algorithm (with the loosened similarity definition as shown in Equation 6.3). Then we experimented with "interesting" page clusters obtained with the original PageGather algorithm (similarity defined in Equation 6.2). By comparing the results of these two kinds of page clusters, we can determine the value of "interesting" page clusters on navigation improvement.

7.4.4.1 Experiment 1: Similarity and Support Settings

We first experimented with page clusters (obtained with generalized PageGather algorithm) generated from different similarity and support settings. While the minimum similarity was adjusted between 5% and 75%, to compare the results the minimum support of page pairs was set accordingly to generate approximately 200 clusters. The results of different kinds of page clusters (content-page-insensitive, content-page-oriented, and content-page-only) are shown respectively in Table 7.12, Table 7.13, and Table 7.14.

Summary of Experiment 1

(1) With our current navigation compression mechanism and the associated parameter settings for evaluation ($r\_limit = 10$, $r\_cost = 0.05$), the overall performance of usage-cluster-based NCMs is much better than that of rule-based NCMs: with a

Table 7.12: Experimental Results for Usage-Cluster-based NCMs (UC-1.1)

content-page-insensitive clusters

RL

Sim.	***Sup.***	#Clusters	Avg. Size	Coverage (total / content)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
5%	*0.116%*	202	2.62	329 / 266	2.98% 5.38% 15.60%	0.40% 3.19% 11.49%	5.86% $(\frac{11162}{190563})$	27.13% $(\frac{11162}{41148})$
25%	*0.085%*	201	2.58	351 / 289	2.50% 4.52% 14.89%	1.24% 3.30% 12.45%	9.88% $(\frac{9249}{93652})$	22.48% $(\frac{9249}{41148})$
50%	*0.044%*	201	2.48	386 / 290	1.28% 2.31% 10.55%	0.81% 1.81% 9.25%	16.26% $(\frac{5660}{34816})$	13.76% $(\frac{5660}{41148})$
75%	*0.021%*	209	3.18	433 / 257	0.50% 0.90% 8.04%	0.33% 0.71% 6.97%	17.14% $(\frac{2161}{12605})$	5.25% $(\frac{2161}{41148})$

MFR

Sim.	***Sup.***	#Clusters	Avg. Size	Coverage (total / content)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
5%	*0.108%*	200	2.69	334 / 226	2.61% 5.03% 14.94%	0.39% 3.01% 11.32%	6.88% $(\frac{11319}{164546})$	30.11% $(\frac{11319}{37598})$
25%	*0.071%*	202	2.69	370 / 245	2.38% 4.59% 15.16%	1.38% 3.47% 12.75%	13.13% $(\frac{9788}{74525})$	26.03% $(\frac{9788}{37598})$
50%	*0.030%*	200	2.79	411 / 259	1.68% 3.24% 12.49%	1.25% 2.71% 11.26%	19.55% $(\frac{6286}{32157})$	16.72% $(\frac{6286}{37598})$
75%	*0.009%*	197	2.88	475 / 264	0.87% 1.67% 11.30%	0.73% 1.51% 10.62%	23.12% $(\frac{2235}{9667})$	5.94% $(\frac{2235}{37598})$

MFR-RL

Sim.	***Sup.***	#Clusters	Avg. Size	Coverage (total / content)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
5%	*0.095%*	200	2.69	352 / 220	2.29% 5.21% 17.21%	0.46% 3.62% 14.15%	5.38% $(\frac{7270}{135045})$	30.36% $(\frac{7270}{23945})$
25%	*0.060%*	200	2.75	388 / 241	2.09% 4.76% 17.52%	1.16% 3.77% 15.12%	9.13% $(\frac{6278}{68732})$	26.22% $(\frac{6278}{23945})$
50%	*0.035%*	200	3.19	419 / 232	1.14% 2.61% 13.07%	0.76% 2.14% 11.77%	12.55% $(\frac{3597}{28664})$	15.02% $(\frac{3597}{23945})$
75%	*0.017%*	218	3.12	500 / 246	0.36% 0.82% 8.49%	0.23% 0.68% 7.79%	10.61% $(\frac{1017}{9586})$	4.25% $(\frac{1017}{23945})$

Table 7.13: Experimental Results for Usage-Cluster-based NCMs (UC-1.2)

content-page-oriented clusters

RL

Sim.	*Sup.*	#Clusters	Avg. Size	Coverage (total / content)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
5%	*0.035%*	201	4.04	261 / 153	2.42% 4.37% 16.05%	1.48% 3.43% 13.67%	12.61% $(\frac{8729}{69247})$	21.21% $(\frac{8729}{41148})$
25%	*0.012%*	195	3.02	357 / 204	1.20% 2.17% 14.86%	0.96% 1.79% 13.16%	24.22% $(\frac{4386}{18109})$	10.66% $(\frac{4386}{41148})$
50%	*0.004%*	245	2.61	449 / 325	0.29% 0.53% 8.22%	0.25% 0.45% 7.53%	37.65% $(\frac{1281}{3402})$	3.11% $(\frac{1281}{41148})$
75%	*0.002%*	190	3.23	473 / 282	0.04% 0.08% 4.43%	0.03% 0.06% 4.10%	23.05% $(\frac{174}{755})$	0.42% $(\frac{174}{41148})$

MFR

Sim.	*Sup.*	#Clusters	Avg. Size	Coverage (total / content)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
5%	*0.032%*	195	4.22	285 / 174	2.63% 5.07% 16.07%	1.65% 3.89% 13.33%	15.12% $(\frac{11017}{72842})$	29.30% $(\frac{11017}{37598})$
25%	*0.016%*	193	3.05	337 / 229	1.97% 3.79% 17.93%	1.62% 3.20% 16.10%	25.35% $(\frac{6496}{25625})$	17.28% $(\frac{6496}{37598})$
50%	*0.007%*	197	3.34	377 / 275	1.19% 2.30% 16.95%	1.07% 2.06% 15.75%	38.47% $(\frac{3637}{9453})$	9.67% $(\frac{3637}{37598})$
75%	*0.002%*	221	2.62	467 / 320	0.19% 0.37% 7.49%	0.17% 0.34% 7.08%	37.77% $(\frac{514}{1361})$	1.37% $(\frac{514}{37598})$

MFR-RL

Sim.	*Sup.*	#Clusters	Avg. Size	Coverage (total / content)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
5%	*0.027%*	200	7.33	205 / 110	1.77% 4.03% 17.56%	1.11% 3.26% 15.21%	11.94% $(\frac{5814}{48684})$	24.28% $(\frac{5814}{23945})$
25%	*0.007%*	196	2.96	335 / 184	1.06% 2.43% 18.82%	0.89% 2.10% 17.45%	20.09% $(\frac{2542}{12651})$	10.62% $(\frac{2542}{23945})$
50%	0.002%	242	2.73	488 / 316	0.24% 0.56% 10.89%	0.21% 0.50% 10.36%	29.14% $(\frac{654}{2244})$	2.73% $(\frac{654}{23945})$
75%	*0.000%*	220	3.32	689 / 325	0.04% 0.08% 11.88%	0.03% 0.06% 11.35%	6.08% $(\frac{44}{724})$	0.18% $(\frac{44}{23945})$

Table 7.14: Experimental Results for Usage-Cluster-based NCMs (UC-1.3)

content-page-only clusters

RL

Sim.	***Sup.***	#Clusters	Avg. Size	Coverage (total / content)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
5%	*0.009%*	210	2.69	360 / 353	1.50% 2.71% 16.99%	0.81% 1.46% 13.92%	9.52% $(\frac{4853}{50975})$	11.79% $(\frac{4853}{41148})$
25%	*0.006%*	203	2.68	349 / 347	1.15% 2.08% 17.50%	0.89% 1.60% 15.49%	18.41% $(\frac{3632}{19732})$	8.83% $(\frac{3632}{41148})$
50%	*0.003%*	246	2.40	435 / 421	0.28% 0.51% 8.17%	0.22% 0.39% 7.26%	28.07% $(\frac{1303}{4642})$	3.17% $(\frac{1303}{41148})$
75%	*0.000%*	227	3.09	643 / 409	0.05% 0.09% 5.06%	0.03% 0.06% 4.41%	15.76% $(\frac{186}{1180})$	0.45% $(\frac{186}{41148})$

MFR

Sim.	***Sup.***	#Clusters	Avg. Size	Coverage (total / content)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
5%	*0.010%*	204	3.25	397 / 389	2.37% 4.56% 22.83%	1.77% 3.40% 19.96%	14.41% $(\frac{6388}{44334})$	16.99% $(\frac{6388}{37598})$
25%	*0.008%*	195	2.99	389 / 386	2.12% 4.08% 21.91%	1.74% 3.36% 19.79%	20.64% $(\frac{5701}{27626})$	15.16% $(\frac{5701}{37598})$
50%	*0.004%*	255	2.94	462 / 456	1.25% 2.41% 10.05%	1.09% 2.11% 17.51%	30.99% $(\frac{3607}{11638})$	9.59% $(\frac{3607}{37598})$
75%	*0.001%*	254	2.49	556 / 473	0.15% 0.29% 9.07%	0.13% 0.26% 8.54%	28.18% $(\frac{388}{1377})$	1.03% $(\frac{388}{37598})$

MFR-RL

Sim.	***Sup.***	#Clusters	Avg. Size	Coverage (total / content)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
5%	*0.005%*	191	2.85	323 / 307	1.44% 3.29% 25.40%	1.05% 2.40% 22.42%	10.26% $(\frac{2964}{28888})$	12.38% $(\frac{2964}{23945})$
25%	*0.003%*	194	3.05	341 / 336	1.00% 2.29% 21.47%	0.82% 1.88% 19.62%	16.78% $(\frac{2263}{13489})$	9.45% $(\frac{2263}{23945})$
50%	*0.001%*	284	2.45	571 / 563	0.25% 0.57% 11.79%	0.21% 0.47% 11.08%	20.89% $(\frac{639}{3059})$	2.67% $(\frac{639}{23945})$
75%	*0.000%*	120	2.81	321 / 227	0.02% 0.05% 14.62%	0.01% 0.03% 14.21%	5.44% $(\frac{26}{478})$	0.11% $(\frac{26}{23945})$

fairly small number of clusters, we can obtain navigation improvements (both NI and NI_c) similar to or even better than those obtained with a much larger number of rules. This is probably because clusters tend to have much larger coverage than rules. For example, the coverage of 200 clusters is larger than that of 3000 rules.

However, we also found that cluster-based NCMs tend to generate much more recommendations than rule-based NCMs. For example, with relatively higher page-pair support and lower intra-cluster similarity, the number of recommendations generated from 200 clusters is larger than the number of recommendations generated from 3000 rules. Such being the case, we can expect that setting a higher recommendation cost (r_cost) will bring more impairment to the NI_c of cluster-based NCMs than to that of rule-based NCMs.

(2) Page clusters generated with higher page-pair support and lower intra-cluster similarity tend to have better results in navigation improvement. We found that although these clusters tend to have lower coverage, they still have the potential to make more recommendations on content pages. The reason why this happens is still unclear.

Also, similar to rule-based NCMs, taking recommendation cost into account can lead to different evaluation results. For example, using content-page-insensitive clusters and RL-based content pages (as shown in Table 7.12), the best $NI(v)$ is obtained with minimum similarity being 5%, but the corresponding $NI_c(v)$ is clearly not the best.

(3) Clusters generated with lower page-pair support and higher intra-cluster similarity tend to result in higher precision but lower recall.

(4) Clusters with very high intra-cluster similarity are generally less useful for navigation improvement, because the number of recommendations generated from such clusters is very limited.

(5) The navigation improvements obtained with different kinds of clusters are also different. Content-page-insensitive clusters tend to have better results in NI, while content-page-oriented and content-page-only clusters tend to have better results in NI_c. Moreover, content-page-oriented and content-page-only clusters tend to result in higher precision but lower recall than content-page-insensitive clusters.

Content-page-oriented and content-page-only clusters are much more cautious about generating recommendations than content-page-insensitive clusters. Therefore, a different r_cost setting will being a much larger impact on the NI_c results of content-page-insensitive clusters than on the other two.

The navigation improvement obtained with content-page-insensitive clusters is less sensitive to support and similarity settings than those obtained with content-page-oriented and content-page-only clusters.

(6) Among the three content page identification methods (RL, MFR, and MFR-RL), MFR tends to have the best overall performance (i.e., potential for best navigation improvement) with usage-cluster-based NCMs. However, the navigation improvement obtained with MFR is not considerably higher than those obtained with the other two methods (RL and MFR-RL).

In addition, the navigation improvement obtained with MFR is less sensitive to the cluster's support and similarity settings no matter which kind of clusters are

Table 7.15: Experimental Results for Maximum Similarity Threshold

content-page-insensitive clusters
(min_support = 0.085% , min_similarity = 25%)

RL-based content pages

Max. Sim.	#Clusters	Avg. Size	Coverage (total / content)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
100%	201	2.58	351 / 289	2.50% 4.52% 14.89%	1.24% 3.30% 12.45%	9.88% $(\frac{9249}{93652})$	22.48% $(\frac{9249}{41148})$
90%	201	2.57	345 / 285	2.50% 4.51% 14.91%	1.23% 3.29% 12.47%	9.88% $(\frac{9235}{93429})$	22.44% $(\frac{9235}{41148})$
75%	209	2.54	336 / 279	2.47% 4.46% 15.17%	1.22% 3.26% 12.77%	9.54% $(\frac{8825}{92464})$	21.45% $(\frac{8825}{41148})$
50%	208	2.32	285 / 236	2.23% 4.03% 16.47%	1.12% 2.97% 14.21%	8.52% $(\frac{6958}{81653})$	16.91% $(\frac{6958}{41148})$

used, while those obtained with RL and MFR-RL are much more sensitive with content-page-oriented and content-page-only clusters.

In the following experiments we tested other features of usage-cluster-based NCMs with content-page-insensitive clusters and RL-based content pages.

7.4.4.2 Experiment 2: Maximum Similarity Threshold

We have found that clusters with very high intra-cluster similarity are generally less useful. Therefore, in this experiment we generated clusters limited to a maximum similarity threshold. Such being the case, the similarity between two pages p_1 and p_2 is computed as:

$$sim(p_1,p_2) = \begin{cases} sim(p_1,p_2), & \text{if } sim(p_1,p_2) \leq \theta \\ 0, & \text{if } sim(p_1,p_2) > \theta \end{cases} \tag{7.3}$$

where θ is a pre-defined maximum similarity. While the maximum similarity threshold was adjusted between 100% and 50%, the minimum support and similarity were fixed: $min\_support = 0.085\%$, $min\_similarity = 25\%$. The results of this experiment are shown in Table 7.15.

The results confirm that clusters with very high intra-cluster similarity are less useful for navigation improvement. For example, by excluding those similarity values higher than 75%, we can still obtain 99% of the navigation improvement. However, the number of pages with very high pair-wise similarity is very limited, which means excluding them can't considerably reduce the size of the cluster set, and therefore can't considerably improve the performance of recommendation.

In the following experiments we tested other features of usage-cluster-based NCMs by restricting the maximum similarity threshold to 75%.

Table 7.16: Experimental Results for "Interesting" Page Clusters

content-page-insensitive clusters
(min_support = 0.085% , min_similarity = 25%, max_similarity = 75%)

RL-based content pages

Cluster Type	#Clusters	Avg. Size	Coverage (total / content)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
all clusters	209	2.54	336 / 279	2.47% 4.46% 15.17%	1.22% 3.26% 12.77%	9.54% $(\frac{8825}{92464})$	21.45% $(\frac{8825}{41148})$
"interesting" clusters	72	2.85	104 / 86	1.53% 2.77% 25.58%	1.16% 2.34% 23.71%	11.15% $(\frac{3054}{27382})$	7.42% $(\frac{3054}{41148})$

7.4.4.3 Experiment 3: "Interesting" Page Clusters

In this experiment we generated only "interesting" clusters (obtained with the original PageGather algorithm) and compared the navigation improvement with that obtained with all the clusters (obtained with the generalized PageGather algorithm). The result of this experiment is shown in Table 7.16.

The result of this experiment is similar to that of the corresponding experiment of rule-based NCMs: (1) recommendations generated from "interesting" clusters have the potential to gain higher navigation improvement than those recommendations generated from uninteresting clusters, if only the correct recommendations have been made; (2) uninteresting clusters are still useful because there is a remarkable portion (more than 38%) of the navigation improvement not covered by interesting clusters, which indicates that even if the relevant page is linked by the current page the user is viewing, sometimes the user may miss these "correct" links.

7.4.4.4 Experiment 4: Varying Number of Page Clusters

In this experiment we tested how navigation is improved with a varying number of usage-based page clusters. Here we set the minimum similarity to 25%, set the maximum similarity to 75%, and adjust only the page-pair support. The results of this experiment are shown in Table 7.17.

The result of this experiment is similar to that of rule-based NCMs: with a given set of content pages and a varying number of clusters, $NI(v_r)$ and $NI_c(v_r)$ do not change much, while $NI(v)$, $NI(v_c)$ and $NI_c(v)$, $NI_c(v_c)$ are improved when the number of page clusters becomes larger, as illustrated in Figure 7.4. Also note that the result pattern of NI_c could be different with different settings of r_cost.

In addition, clusters with higher page-pair support (e.g., $support \geq 0.041\%$) tend to be more useful for navigation improvement, while clusters with lower support (e.g., $support \leq 0.033\%$) tend to be less useful. The results also show that the precision becomes lower with more clusters (lower page-pair support), while the recall becomes higher.

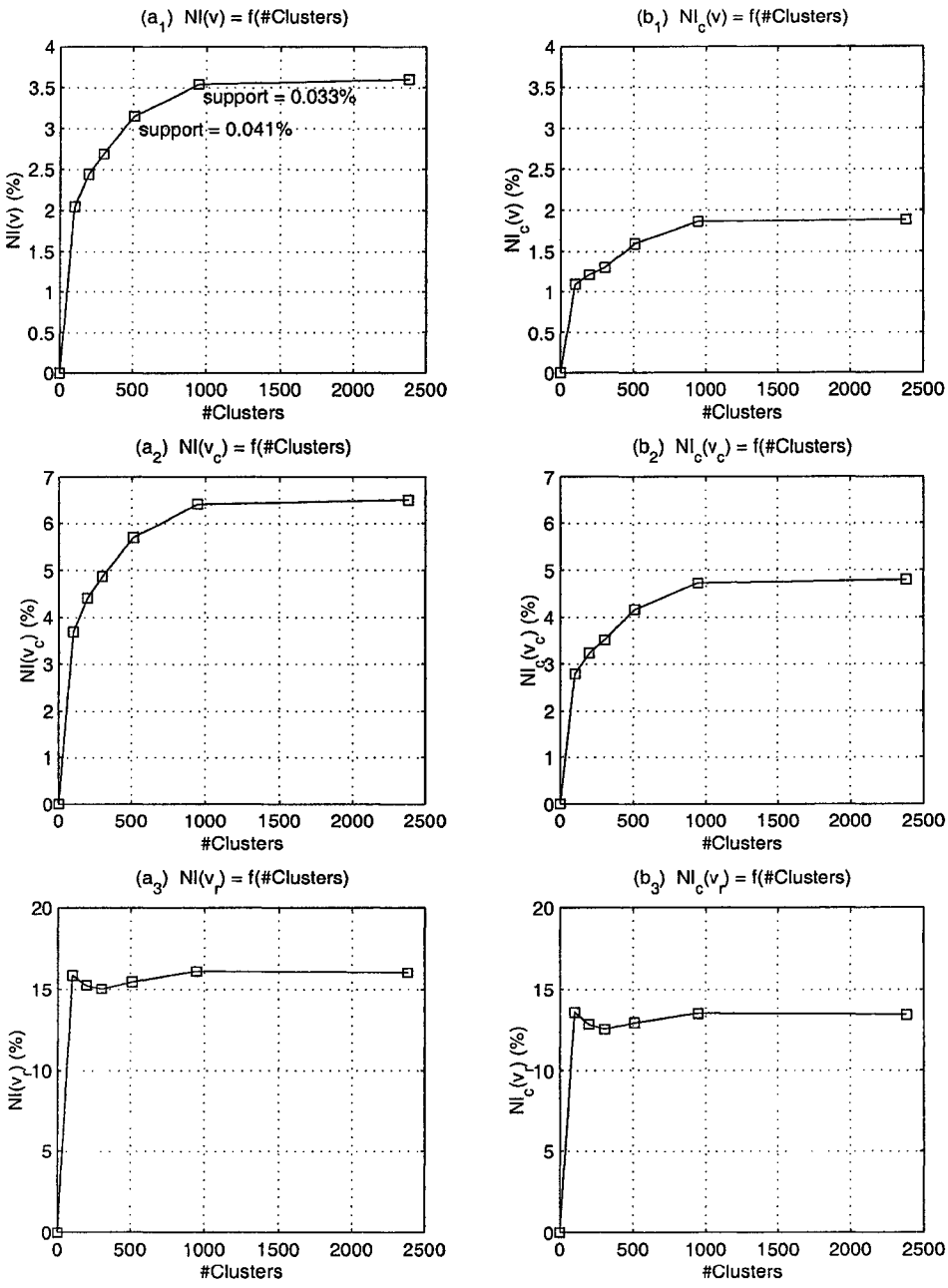


Figure 7.4: Navigation Improvement vs. #Clusters

Table 7.17: Experimental Results for Varying Number of Page Clusters

content-page-insensitive clusters
(min_similarity = 25%, max_similarity = 75%)

RL-based content pages

Support	#Clusters	Avg. Size	Coverage (total / content)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
0.167%	101	2.66	172 / 139	2.04% 3.68% 15.85%	1.09% 2.78% 13.56%	10.36% $(\frac{7246}{69934})$	17.61% $(\frac{7246}{41148})$
0.090%	200	2.54	318 / 265	2.44% 4.41% 15.24%	1.21% 3.23% 12.84%	9.56% $(\frac{8695}{90976})$	21.13% $(\frac{8695}{41148})$
0.056%	303	2.56	492 / 399	2.69% 4.87% 15.02%	1.30% 3.51% 12.56%	9.31% $(\frac{9631}{103473})$	23.41% $(\frac{9631}{41148})$
0.041%	509	2.59	745 / 593	3.15% 5.69% 15.46%	1.59% 4.16% 12.95%	9.48% $(\frac{10942}{115425})$	26.59% $(\frac{10942}{41148})$
0.033%	944	3.24	945 / 728	3.54% 6.41% 16.09%	1.86% 4.73% 13.50%	9.50% $(\frac{11865}{124955})$	28.83% $(\frac{11865}{41148})$
0.030%	2382	4.94	1023 / 778	3.60% 6.50% 16.02%	1.88% 4.80% 13.42%	9.43% $(\frac{11982}{127021})$	29.12% $(\frac{11982}{41148})$

7.4.4.5 Experiment 5: More Training and Test Data

In this experiment we tested the navigation improvement of applying usage-cluster-based NCMs on more log data with RL-based content pages and content-page-insensitive clusters. We used the same data set (and the same content page set) used for testing rule-based NCMs. And we also assumed that the knowledge was updated on a monthly basis. Moreover, we assumed that our recommendation system could handle 1000 clusters for real-time response. Therefore, in each case we only generated approximately 1000 clusters. The results of this experiment are shown in Table 7.18 and Table 7.19.

The results show that the navigation improvements (both NI and NI_c) obtained with 1000 usage-cluster-based NCMs are constantly better than those obtained with 2000 rule-based NCMs. Compared with the recommendations generated with rule-based NCMs, those recommendations generated with usage-cluster-based NCMs tend to have similar precision but much higher recall.

However, the observation that cluster-based NCMs tend to generate much more recommendations than rule-based NCMs implies that the comparison between their NI_c results is inevitably dependent on the choice of the r_cost parameter.

7.4.4.6 Summary

- Clusters with higher page-pair support tend to be more useful for navigation improvement, while clusters with very high intra-cluster similarity (e.g., higher than 75%) are generally less useful.

Table 7.18: Experimental Results on More UACS Log Data

content-page-insensitive clusters
(min_similarity = 25%, max_similarity = 75%)
(page-pair support adjusted to generate approximately 1000 page clusters)

RL-based content pages
(support = 2, confidence = 5%)

Training Data	Test Data	#Clusters	#Content Pages (training / test)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
01/2001	02/2001	972	7938 / 7591	4.10% 5.90% 12.67%	1.55% 3.62% 9.54%	9.23% $(\frac{16185}{175346})$	26.14% $(\frac{16185}{61921})$
02/2001	03/2001	1025	7591 / 8554	2.90% 4.30% 10.46%	0.82% 2.45% 7.68%	9.38% $(\frac{14383}{153319})$	21.12% $(\frac{14383}{68089})$
03/2001	04/2001	1077	8554 / 8675	3.03% 4.13% 10.30%	1.26% 2.48% 7.73%	12.03% $(\frac{15546}{129262})$	19.00% $(\frac{15546}{81838})$
04/2001	05/2001	1010	8675 / 8906	2.73% 3.65% 10.58%	1.19% 2.15% 7.73%	14.34% $(\frac{13703}{95588})$	20.49% $(\frac{13703}{66890})$
05/2001	06/2001	1008	8906 / 7855	3.16% 4.16% 10.60%	1.42% 2.44% 7.79%	12.73% $(\frac{9915}{77867})$	19.58% $(\frac{9915}{50646})$
06/2001	07/2001	984	7855 / 8085	2.75% 3.50% 9.35%	1.21% 1.94% 6.72%	13.73% $(\frac{10091}{73518})$	17.51% $(\frac{10091}{57633})$
07/2001	08/2001	972	8085 / 9368	3.20% 4.10% 9.62%	1.32% 2.22% 6.72%	12.59% $(\frac{13578}{107822})$	19.38% $(\frac{13578}{70066})$
08/2001	09/2001	1084	9368 / 9096	2.49% 3.47% 10.04%	0.28% 1.53% 7.01%	7.80% $(\frac{15695}{201223})$	18.62% $(\frac{15695}{84281})$
09/2001	10/2001	953	9096 / 10291	2.55% 3.73% 11.32%	0.53% 2.06% 8.58%	7.59% $(\frac{16907}{222789})$	18.12% $(\frac{16907}{93329})$
10/2001	11/2001	1060	10291 / 10280	2.30% 3.34% 10.21%	0.63% 1.81% 7.68%	9.30% $(\frac{14514}{156014})$	17.41% $(\frac{14514}{83352})$
11/2001	12/2001	1021	10280 / 9550	3.07% 4.31% 11.12%	1.34% 2.64% 8.54%	10.62% $(\frac{14199}{133738})$	20.52% $(\frac{14199}{69184})$
12/2001	01/2002	1033	9550 / 10616	2.50% 3.55% 10.68%	0.88% 1.99% 7.97%	9.24% $(\frac{15179}{164228})$	17.01% $(\frac{15179}{89244})$

Table 7.19: Experimental Results on More UACS Log Data (Continued)

content-page-insensitive clusters
(min_similarity = 25%, max_similarity = 75%)
(page-pair support adjusted to generate approximately 1000 page clusters)

RL-based content pages
(support = 2, confidence = 5%)

Training Data	Test Data	#Clusters	#Content Pages (training / test)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
01/2002	02/2002	1062	10616 / 9730	2.72% 4.63% 11.32%	1.03% 2.85% 8.66%	10.07% $(\frac{16375}{162642})$	22.95% $(\frac{16375}{71342})$
02/2002	03/2002	995	9730 / 10440	2.80% 4.05% 11.57%	1.21% 2.56% 9.07%	10.81% $(\frac{15716}{145450})$	18.70% $(\frac{15716}{84043})$
03/2002	04/2002	1006	10440 / 11059	2.37% 3.40% 10.57%	1.04% 2.14% 8.27%	11.53% $(\frac{13426}{116488})$	15.53% $(\frac{13426}{86471})$
04/2002	05/2002	1033	11059 / 10368	2.58% 3.46% 10.80%	1.25% 2.16% 8.30%	13.91% $(\frac{13454}{96692})$	15.95% $(\frac{13454}{84370})$
05/2002	06/2002	1032	10368 / 9813	3.01% 4.56% 12.08%	1.59% 2.98% 9.07%	15.73% $(\frac{15172}{96475})$	21.04% $(\frac{15172}{72094})$
06/2002	07/2002	1006	9813 / 9420	3.29% 4.46% 12.81%	1.85% 2.99% 10.00%	14.13% $(\frac{14059}{99512})$	19.12% $(\frac{14059}{73542})$
07/2002	08/2002	1013	9420 / 6333	3.27% 5.01% 14.52%	1.68% 3.24% 11.22%	10.97% $(\frac{11355}{103556})$	22.83% $(\frac{11355}{49729})$
08/2002	09/2002	972	6333 / 6783	2.23% 3.62% 13.50%	0.73% 2.17% 10.30%	8.26% $(\frac{12187}{147480})$	18.10% $(\frac{12187}{67325})$
09/2002	10/2002	1058	6783 / 7281	3.12% 4.96% 13.91%	1.39% 3.35% 11.11%	9.59% $(\frac{16003}{166850})$	24.39% $(\frac{16003}{65610})$
10/2002	11/2002	1020	7281 / 6953	2.83% 4.49% 14.10%	1.52% 3.20% 11.58%	11.41% $(\frac{12385}{108560})$	21.85% $(\frac{12385}{56682})$
11/2002	12/2002	1029	6953 / 6812	1.66% 2.77% 12.35%	0.88% 1.93% 10.19%	11.74% $(\frac{7229}{61600})$	13.86% $(\frac{7229}{52139})$
12/2002	01/2003	1007	6812 / 6369	2.54% 3.99% 12.97%	1.09% 2.60% 10.56%	8.88% $(\frac{11491}{129436})$	18.67% $(\frac{11491}{61536})$
01/2003	02/2003	1011	6369 / 5983	3.58% 5.85% 15.68%	1.82% 4.22% 13.02%	9.46% $(\frac{12349}{130588})$	25.57% $(\frac{12349}{48304})$

- Content-page-insensitive clusters tend to have the best results in NI, and are generally less sensitive to different page-pair support and intra-cluster similarity settings. However, content-page-oriented and content-page-only clusters are much more cautious about generating recommendations. Therefore, a different setting of r_cost will have a much larger impact on content-page-insensitive clusters than on the other two.
- We can argue that with the heuristics and parameter settings used in these experiments, usage-cluster-based NCMs are more suitable than rule-based NCMs for the application of improving user navigation by dynamic recommendations. However, using different heuristics and parameter settings can certainly lead to different results. For example, we found that cluster-based NCMs tend to generate much more recommendations than rule-based NCMs. Therefore, setting a higher recommendation cost (r_cost) will bring much more impairment to the NI_c of cluster-based NCMs than to that of rule-based NCMs.

7.4.5 Content-based Clusters

Discovering purely content-based page clusters is itself a very difficult challenge. In our experiments we generated content-based page clusters using CBC algorithm. The CBC algorithm itself involves a large number of parameters, many of which should be determined empirically. To simplify the problem, here we generated only one set of content-based page clusters using a set of default settings. For more information about the CBC algorithm, please refer to [53].

By applying the CBC algorithm (with its default settings) to the training data set — the UACS web logs of January 2003, we generated 902 content-based page clusters with a total coverage of 27575 distinct pages. As previously discussed, our content-cluster-based NCMs are obtained by matching these purely content-based clusters with a set of pre-generated usage-based clusters. This is accomplished within the following three steps:

1. Generate a set of usage-based page clusters. Since these usage-based clusters are being used for matching content-based clusters, they have to be content-page-insensitive.[2] For simplicity we generated only one set of usage-based clusters with RL-based content pages. The support and confidence of content pages are set to 2 and 5% respectively. Moreover, the minimum page-pair support is set to 0.033%, the minimum intra-cluster similarity is set to 25%, and the maximum intra-cluster similarity is set to 75%. The total number of usage-based clusters generated in this step is 1011.
2. Refine the original content-based page clusters as follows: for each cluster, remove all its members whose similarity to the centroid of the cluster is lower than a pre-defined threshold. This restriction is applied to maintain the quality of the content-based page clusters.
   Besides, since the recommendations are restricted to content pages only, and the content page set used for recommendations has been pre-determined, we can eliminate those content-based page clusters that do not contain such content pages.

[2] All content-based page clusters are content-page-insensitive.

3. For each refined content-based cluster, find its matching usage-based cluster. The quality of this matching is controlled by a pre-defined minimum matching similarity (as defined in section 6.3.2.2). If the matching usage-based cluster is found, then this refined content-based cluster will be included in our content-cluster-based NCMs.

This procedure involves two important parameters: one is the minimum similarity between cluster members and the cluster centroid (for content-based clusters only), denoted as θ; another one is the minimum matching similarity between usage-based clusters and content-based clusters, denoted as ε. Different settings of these two parameters will lead to different results. Our experiments showed that a relatively better result (considering both NI and NI_c) is obtained with $\theta = 35\%$ and $\varepsilon = 15\%$.

With $\theta = 35\%$ we obtained a set of 537 refined content-based clusters. This cluster set has an average size of 34.76 and a total coverage of 16630 distinct pages, in which 3085 pages are content pages. Then we applied the matching process with $\varepsilon = 15\%$ to obtain the content-cluster-based NCMs. In this process, only 14% of the refined content-based clusters and 17% of the usage-based clusters are matched, and the average matching similarity is less than 25%.

The experimental results (on navigation improvement) of the content-cluster-based NCMs and the corresponding usage-cluster-based NCMs are shown in Table 7.20. The results show that the overall performance of content-cluster-based NCMs is much worse than that of usage-cluster-based NCMs, therefore may not be appropriate for improving user navigation by dynamic recommendations. This is because content-based clusters tend to have much larger sizes, which means they tend to generate much more recommendations, and their precision of recommendations will be much lower than that of usage-based clusters. Consequently, their recommendation cost will be much higher. As shown in Table 7.20, the NI_c obtained with content-cluster-based NCMs is extremely low.

If we set θ or ε too high, the final content-based clusters will have a much smaller coverage, which leads to a much smaller NI. On the other hand, if we set θ or ε too low, the final content-based clusters will be much larger in size, which leads to a much smaller NI_c.

7.5 More Data: The "Music Machines" Web Logs

In addition to the UACS web logs, we also experimented with another data set — a set of publicly available server access logs of the music machines web site (currently at "`http://machines.hyperreal.org/`"). These web logs are used by the "adaptive web sites" project [56, 57] (Perkowitz and Etzioni, Department of Computer Science and Engineering, University of Washington), and made available online at "`http://www.cs.washington.edu/ai/adaptive-data/`".

The music machines web site provides a wide variety of information related to music equipments, including images, softwares, schematics, as well as tips and comments from musicians. In this experiment, we have collected seven months of access logs from the music machines web site, from October 1, 1998 to April 30, 1999.

The access log format used by the music machines web site is a standard log format for HTTP servers. A big difference between this log format and the one used for the UACS web logs is that this log format does not contain cookies. Therefore, the

Table 7.20: Experimental Results for Content-based Page Clusters

CBC-generated content-based clusters
(min_similarity_to_centroid = 35%)
(min_matching_similarity_with_usage-based_clusters = 15%)

content-page-insensitive usage-based clusters
(min_support = 0.033% , min_similarity = 25%, max_similarity = 75%)

RL-based content pages
(support = 2, confidence = 5%)
(#content pages: training set = 6369, test set = 5983)

Cluster Type	#Clusters	Avg. Size	Coverage (total / content)	NI ($v/v_c/v_r$)	NI_c ($v/v_c/v_r$)	Precision	Recall
usage-based clusters	1011	3.21	987 / 822	3.58% 5.85% 15.68%	1.82% 4.22% 13.02%	9.46% $(\frac{12349}{130588})$	25.57% $(\frac{12349}{48304})$
content-based clusters	73	13.53	968 / 489	1.33% 2.17% 15.61%	-0.88% 0.13% 10.28%	3.01% $(\frac{4925}{163585})$	10.20% $(\frac{4925}{48304})$

cookie-only method is not applicable to this data set, which makes ip-link method the only choice for user identification. For session identification, we still use the timeout-based method with the timeout set to 30 minutes.

Our data preparation experiments showed that for each month of this data set, the music machines web server generates approximately one million records and 40000 useful sessions (1<session length$\leq$100). The average session length of the useful sessions is approximately 7.43, which is slightly longer than that of the UACS data set (shorter than 6.0). Moreover, the music machines web logs involve a much higher revisitation rate than the UACS web logs.

> Given a set of web logs, suppose the number of distinct web pages (in our case, static HTML pages) is N_p, and the number of records generated from the requests of these web pages is N_r, then the *revisitation rate* of this web log set is: $\frac{N_r}{N_p}$.

The monthly music machines web logs include less than 3000 distinct static HTML pages with no less than 250000 records requesting these pages, while the monthly UACS web logs include as least 40000 distinct static HTML pages with no more than 800000 records requesting these pages. Provided with the above definition, the revisitation rate of the music machines web logs is at least $\frac{250000}{3000} \approx 83$, and the revisitation rate of the UACS web logs is at most $\frac{800000}{40000} = 20$. Therefore, the revisitation rate of the music machines web log data is at least four times higher than that of the UACS web log data.

For content page identification, we use the Reference Length method. By applying the clustering algorithm described in Section 5.3.1.1, the obtained cutoff viewing time is 24.07 seconds ($\sigma = 0.34$).

We report experiments on the music machines web log data with two kinds of NCMs, including NCMs obtained from association rules and usage-based page clusters. The experiments showed that these NCMs possess features similar to those

NCMs obtained from the UACS web log data. The results of these experiments are shown in Table 7.21. Note that different data sets may have different characteristics. Therefore, to achieve better results, the parameters used to learn the same kind of NCMs could be set differently on different data sets.

The results show that the navigation improvements (both NI and NI_c) we can expect from the music machines web log data are much higher than those we can expect from the UACS web log data, even with a much smaller number of NCMs (e.g., see the results of usage-cluster-based NCMs). There are two possible reasons for this:

- As mentioned above, the music machines web logs involve a much higher revisitation rate than the UACS web logs, therefore may have better potential for navigation improvement.
- In the music machines web log data, the proportion of sessions in which the users did not visit any content page is much smaller than that of the UACS web log data. This proportion is generally larger than 50% in the UACS web logs but lower than 15% in the music machines web logs. This means that most users to the music machines web site are target-driven, which gives their visits potential for navigation improvement.

The results from different monthly data sets are quite consistent, except that when using the March 1999 web logs as training set and April 1999 web logs as test set, the result is much worse than those obtained with other monthly data sets. The reason why this happens is still unclear.

This experiment clearly demonstrates the important role of data in the application of improving web navigation: that different data sets may possess different characteristics, and therefore have different potentials for navigation improvement.

7.6 Annotated Data: The "Travel Study" Web Logs

Another data set we used in our experiments is a set of "travel study" web logs that was collected by Zhu et. al in 2002 and used in their WebIC project [82, 84, 83] (Zhu, Greiner, Department of Computing Science; Häubl, School of Business; University of Alberta).

These web logs were collected through a specially designed experiment. The participants were 144 undergraduate students from School of Business at the University of Alberta. Each student was given about 45 minutes to perform a specific task: identify three different vacation places that the participant has never been to before, and make a detailed plan for each vacation place, including travel dates, flights, accommodation, activities, etc. The participants were asked to use an enhanced browsing tool AIE (Annotated Internet Explorer [84]), with which they could easily label the "important" pages (content pages) and produce a brief report summarizing their vacation plans. Due to some technical problems, only 114 participants' web logs were used in our experiments.

These travel study web logs are different from the UACS web logs and the Music Machines web logs in the following aspects:

- These web logs were collected at the client side. The user visits were not restricted to a single web site, and the valid URLs were not restricted to static

Table 7.21: Experimental Results on Music Machines Log Data

RL-based content pages
(support = 2, confidence = 5%)

content-page-insensitive association rules
(min_confidence = 25%, max_confidence = 90%)
(support adjusted to generate approximately 2000 association rules)

Training Data	Test Data	#Rules	#Content Pages (training / test)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
10/1998	11/1998	2008	891 / 1174	4.50% 4.86% 10.40%	2.47% 2.88% 8.23%	9.99% $(\frac{10621}{106358})$	16.08% $(\frac{10621}{66071})$
11/1998	12/1998	2015	1174 / 1215	4.33% 4.69% 9.86%	2.18% 2.60% 7.60%	9.29% $(\frac{10365}{111566})$	16.22% $(\frac{10365}{63908})$
12/1998	01/1999	2057	1215 / 1300	4.09% 4.43% 9.87%	1.99% 2.39% 7.67%	8.67% $(\frac{10374}{119597})$	14.89% $(\frac{10374}{69661})$
01/1999	02/1999	2077	1300 / 1281	3.86% 4.19% 9.13%	1.92% 2.31% 7.15%	9.13% $(\frac{9442}{103412})$	14.71% $(\frac{9442}{64172})$
02/1999	03/1999	2062	1281 / 1346	4.32% 4.73% 10.50%	2.16% 2.63% 8.26%	8.41% $(\frac{10189}{121217})$	15.05% $(\frac{10189}{67718})$
03/1999	04/1999	2045	1346 / 1254	1.12% 1.23% 7.32%	0.32% 0.47% 6.12%	6.59% $(\frac{2670}{40451})$	4.52% $(\frac{2670}{59130})$

content-page-insensitive clusters
(min_similarity = 15%, max_similarity = 75%)
(page-pair support adjusted to generate approximately 500 page clusters)

Training Data	Test Data	#Clusters	#Content Pages (training / test)	NI $(v/v_c/v_r)$	NI_c $(v/v_c/v_r)$	Precision	Recall
10/1998	11/1998	491	891 / 1174	6.94% 7.50% 11.96%	3.48% 4.14% 8.81%	8.45% $(\frac{15347}{181566})$	23.23% $(\frac{15347}{66071})$
11/1998	12/1998	498	1174 / 1215	7.01% 7.60% 11.80%	3.50% 4.21% 8.66%	8.41% $(\frac{15342}{182401})$	24.01% $(\frac{15342}{63908})$
12/1998	01/1999	535	1215 / 1300	6.73% 7.29% 11.75%	3.20% 3.88% 8.64%	7.93% $(\frac{15973}{201508})$	22.93% $(\frac{15973}{69661})$
01/1999	02/1999	509	1300 / 1281	6.56% 7.12% 11.48%	3.27% 3.95% 8.58%	8.07% $(\frac{14169}{175526})$	22.08% $(\frac{14169}{64172})$
02/1999	03/1999	494	1281 / 1346	6.76% 7.39% 11.73%	3.07% 3.85% 8.49%	7.71% $(\frac{15965}{206955})$	23.58% $(\frac{15965}{67718})$
03/1999	04/1999	495	1346 / 1254	4.47% 4.90% 10.36%	1.59% 2.12% 7.80%	5.88% $(\frac{8565}{145754})$	14.49% $(\frac{8565}{59130})$

Table 7.22: Verification of Content Page Identification Methods

Content Page Identification Methods	Precision	Recall
RL (cutoff viewing time = 20.46 seconds)	40%	73%
basic MFR	31%	50%
refined MFR (min. viewing time = 5 seconds)	39%	52%
MFR-RL (min. viewing time = 20.46 seconds)	56%	42%

HTML pages only, i.e., log records from both static and dynamic HTML pages will be used.

- All the users are goal-directed and all the "true" content pages have been labeled by the users.

An important characteristic of this data set is that these web logs involve a much lower revisitation rate (see Page 111 for the definition of revisitation rate) compared with the UACS web logs and the Music Machines web logs. These web logs include 6490 distinct URLs with only 13235 records. Therefore, the revisitation rate of this data set is only $\frac{13235}{6490} \approx 2$, which is 10 times lower than the UACS web log data, and 40 times lower than the Music Machines web log data.

First we used this data set to verify the content page identification methods described in Section 5.3.1. The results are shown in Table 7.22. Though due to some technical problems, the recorded viewing time of the user-labeled "important" pages was not perfectly accurate, it still shows that these heuristic content page identification methods make sense.

Since the web logs were collected at the client side and the users were not restricted to a particular web site in order to achieve a specific task, we can no long determine user sessions based on user navigation behaviors of entering and exiting a web site. Here we address this problem with two approaches. The first approach simply puts all the log records from an individual user into a single session, because each user was given a clear task to fulfill. The sessions identified using this approach are called *task-directed sessions*. However, the problem with this approach is that those sessions identified this way are often too long (the average session length is approximately 116). Therefore, we also use another approach, which assumes that each session ends when the user reaches an "important" page. The sessions identified using this approach are called *content-page-directed sessions*. The idea of this approach is similar to that of server-side transaction identification (see Page 17). The user sessions identified this way are much shorter (the average session length is approximately 24).

We report experiments on this data set with two kinds of content-page-oriented NCMs[3], including association-rule-based NCMs and usage-cluster-based NCMs. In each case, we ran a 10-fold cross validation [46] on the entire session repository. Note that we adjusted the parameters of the NCM learning algorithms if necessary to achieve better results. The results of these experiments are summarized as follows:

- The navigation improvement we can expect from task-directed sessions is very

[3]Our experiments showed that with this particular data set, content-page-oriented NCMs perform much better than content-page-insensitive NCMs.

small. Association-rule-based NCMs and usage-cluster-based NCMs have similar performance on these sessions, and the average navigation improvement ($\overline{NI}$) is only 0.7% ($\sigma = 1.51\%$).

- The navigation improvement we can obtain from content-page-directed sessions is higher than that from task-directed sessions. However, association-rule-based NCMs and usage-cluster-based NCMs have completely different performance on these sessions. We can obtain no improvement with up to 3000 association rules, but usage-cluster-based NCMs can bring about an average improvement ($\overline{NI}$) of 1.76% ($\sigma = 1.90\%$).

The results show that the navigation improvement (NI) we can expect from this travel study web log data is relatively small, considering that all the user sessions are goal-directed, i.e., each session contains at least one content page. The UACS web log data also bears a low NI, but it is partially because that a significant proportion (more than 50%) of the sessions contains no content page.

The reason we can not obtain a higher NI from this travel study data set is probably that: the duration of the "travel study" experiment was so short that the revisitation rate of this web log data is extremely low. We found that in most cases, no more than 10% of the content pages in the test set also appear as content pages in the training set. Consequently, the recall of the recommendation is generally less than 2%.

This experiment clearly demonstrates that our document-based NCMs can predict only those web pages that *have been previously visited with a certain amount of revisitation.* Such NCMs perform poorly when the revisitation rate is low. In such case, we have to use feature-based methods (including document features and word features; see [82, 84, 83] for details) instead.

7.7 Examples of WebKIV Visualizations

Here we present four visualization examples using our WebKIV tool (see Section 3.4), including the visualizations for web site structure, web surfing animation, web page usage vs. hyperlink usage, and NCM application. Most data used in these experiments is obtained from the Music Machines web logs (see Section 7.5) in January 1999.

For more discussion and experiments about WebKIV, please refer to [50, 49].

7.7.1 Visualization of Web Site Structure

The structure of the Music Machines web site is shown in Figure 7.5, where each node represents a web page and each line represents a hyperlink between two pages. Note that only those lines following the hierarchy structure (i.e., between "parent" nodes and "son" nodes) are displayed.

The visualization of the web site structure has many potential applications on its own. For example, if a branch (or a subtree) is much deeper than other branches, it may indicate that some of the information under this branch is too far away from the web site's home page, and the users will need more steps to reach that information. The web site structure can also reveal the web site's "balance", e.g., the span of one branch is very wide while other branches have only few descendants.

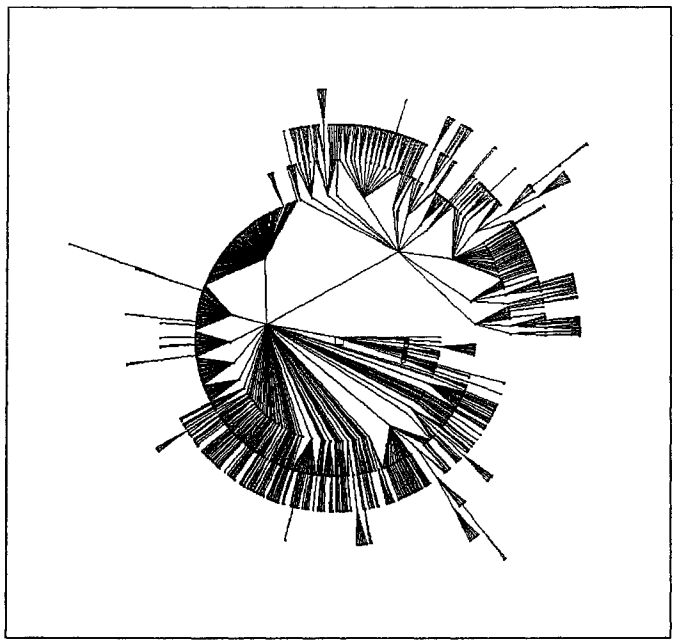


Figure 7.5: Visualization of Music Machines Web Site Structure

7.7.2 Visualization of Web Surfing Animation

An example of web surfing animation is shown by the four static snapshots in Figure 7.6. Each black dot represents an individual surfer in the web site. A dot moving from one place to another indicates the traversal path the surfer followed. A dot stopping at a tree node means that the surfer is viewing the specific web page, and the time a dot stays at a node indicates the time the surfer spends on that web page. Also, a dot will disappear if the user has idled for a certain amount of time (base on the session timeout). WebKIV maintains a count for each hyperlink traversal by all users. A gray scaled line is drawn for each hyperlink based on this count. The more a hyperlink is traversed, the darker its corresponding line will be.

The data of this example is obtained from the Music Machines web logs on January 1, 1999, starting from midnight. The frame refresh rate is set to 10 frames per second, and the animation rate is set 50 times faster than the real time, i.e., one animation second equals 50 seconds in real time.

Figure 7.6 (a) shows a static view of web users' activities at midnight, when there were only several surfers browsing the web site, and no obvious traversal patterns available. The web site became busier in the morning, with around 20 surfers wandering in it. Several heavily traversed hyperlinks started to "stand out", as shown in Figure 7.6 (b). The most crowded time is around evening, with about 30 people visiting the web site. The "hot" web pages and "heavily" used hyperlinks are clearly revealed, as shown in Figure 7.6 (c). Finally, Figure 7.6 (d) shows the one-day trace history left by the web users.

Compared with static web visualization tools, WebKIV can not only show the aggregate web usage during a certain period of time, but also show how the web site is being used over time.

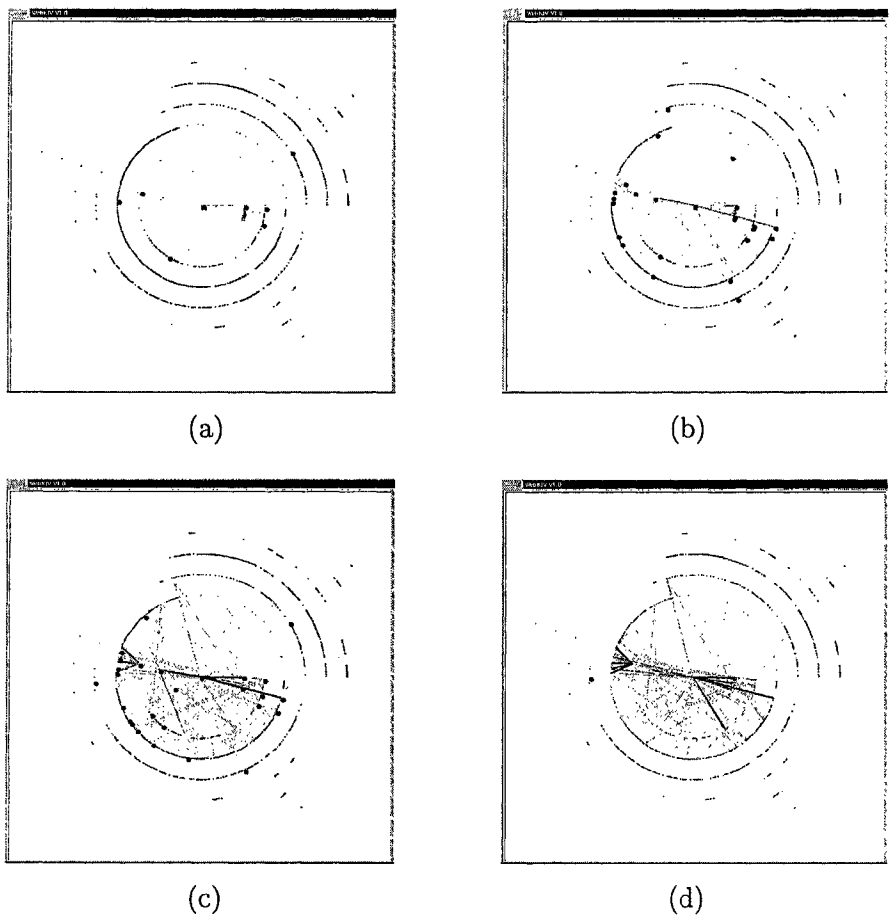
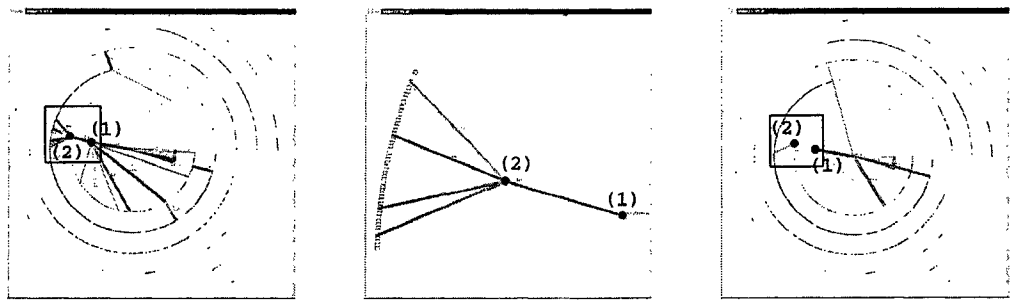


Figure 7.6: Visualization of Web Surfing Animation



(a) (b) (c)

(1) `manufacture.html` (2) `samples.html`

Figure 7.7: Visualization of Web Page Usage vs. Hyperlink Usage

7.7.3 Visualization of Web Page Usage vs. Hyperlink Usage

Note that some patterns can not be easily identified without viewing the data from different perspectives. In the example of Figure 7.7, we are looking for answers to the following questions: (1) "Which web pages did the users visit most?", (2) "Which hyperlinks were used most?", and (3) "From where did the users reach these web pages?"

The data of this example is obtained from the Music Machines web logs in January 1999. Figure 7.7 (a) shows the statistics of web page usage, where each line represents the aggregate usage of its corresponding "son" node. More specifically, the color and thickness of the line indicate the number of times the "son" web page was visited during a certain period of time. Note that a line is displayed only when the visit count of the "son" web page is higher than a pre-defined threshold. As we can see, the web page "`samples.html`" was well visited (The actual number is 660 times. See Figure 7.7 (b) for an enlarged view). However, we can not tell which routes the users followed to reach this page merely from Figure 7.7 (a) itself.

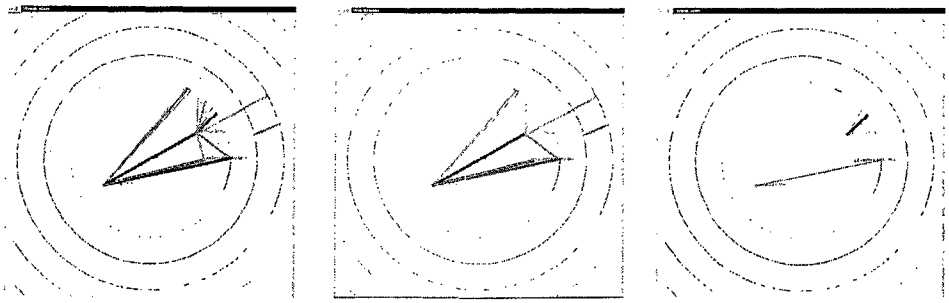
Figure 7.7 (c) shows the statistics of hyperlink usage, where each line represents the aggregate usage of its corresponding hyperlink. Here we found that the users were not browsing via the hierarchical structure. As we can see, the hyperlink from "`manufacture.html`" to "`samples.html`" was rarely used (The actual number is 31 times. Since this count is below a pre-defined threshold, the corresponding line is not shown in the diagram). This implies that the web page "`samples.html`" was mostly accessed either through some other hyperlinks not conforming to the web site's hierarchy structure, or through other related web sites, or through the users' bookmarks directly.

This example demonstrates that we can combine the visualizations of both web page usage and hyperlink usage to help find out the answer for question (3).

7.7.4 Visualization of NCM Application

Our visualization tool WebKIV provides a way of comparing the navigation behaviors of two different navigation strategies, by superimposing two sets of user navigation paths (in our case, the original and the NCM-employed compressed web logs).

The data of this example is obtained from the UACS web logs in October 2001.



(a) before applying NCM (b) after applying NCM (c) pattern (b) - (a)

Figure 7.8: Visualization of NCM Application

As illustrated in Figure 7.8, part (a) is the visualization of a web usage log *before* applying the NCMs, part (b) is a visualization of the same web usage log *after* the NCMs have been applied, and part (c) is a "subtraction" of (b) - (a), which provides a visualization of the difference between the unimproved and improved navigation paths.

Note that in Figure 7.8 (a) and Figure 7.8 (b), a line is displayed only when the traversal count of the corresponding hyperlink is higher than a pre-defined threshold. In Figure 7.8 (c), the red lines represent the NCM-suggested links, and the blue lines are "saved" links.

Chapter 8

A NCM-Based Recommendation System

8.1 Introduction

Here we present an online dynamic recommendation system based on NCMs. As previously discussed, NCMs are knowledge about user navigation patterns that are generated with various learning methods applied to different kinds of web data (e.g., web usage logs, web content, etc.), and can then be used to predict users' interests based on their current navigation paths.

We present a dynamic recommendation system which is intended to help users reach contents of interest more quickly by making dynamic recommendations based on NCMs. As mentioned earlier, this process is called Navigation Compression.

Our recommendation system consists of five major components: (1) User Tracking, (2) Session Management, (3) Footer Generation, (4) Recommendation Generation, and (5) Feedback Management. In the following section we describe the architecture of the system as well as details of the five components.

8.2 System Architecture

The basic architecture of our NCM-based dynamic recommendation system is illustrated by the diagram of Figure 8.1. With respect to this architecture, the procedure of processing one user request can be described as follows:

1. Client sends the request to server. This request can carry three kinds of cookies:
   - *Apache* This cookie is used for user tracking.
   - *SESSION_ID* This cookie is used for session tracking.
   - *USE* This cookie is used for user feedback gathering.
2. Server calls the *User Tracking* module which uses the *Apache* cookie to identify individual users. If the client request does not carry this cookie, a new identification will be assigned to the user and saved under this cookie name.
3. Server retrieves the requested document. If the requested document is an HTML page, then the *Footer Generation* module is called to append a script module (named "Footer", written in JavaScript) into the original HTML page.

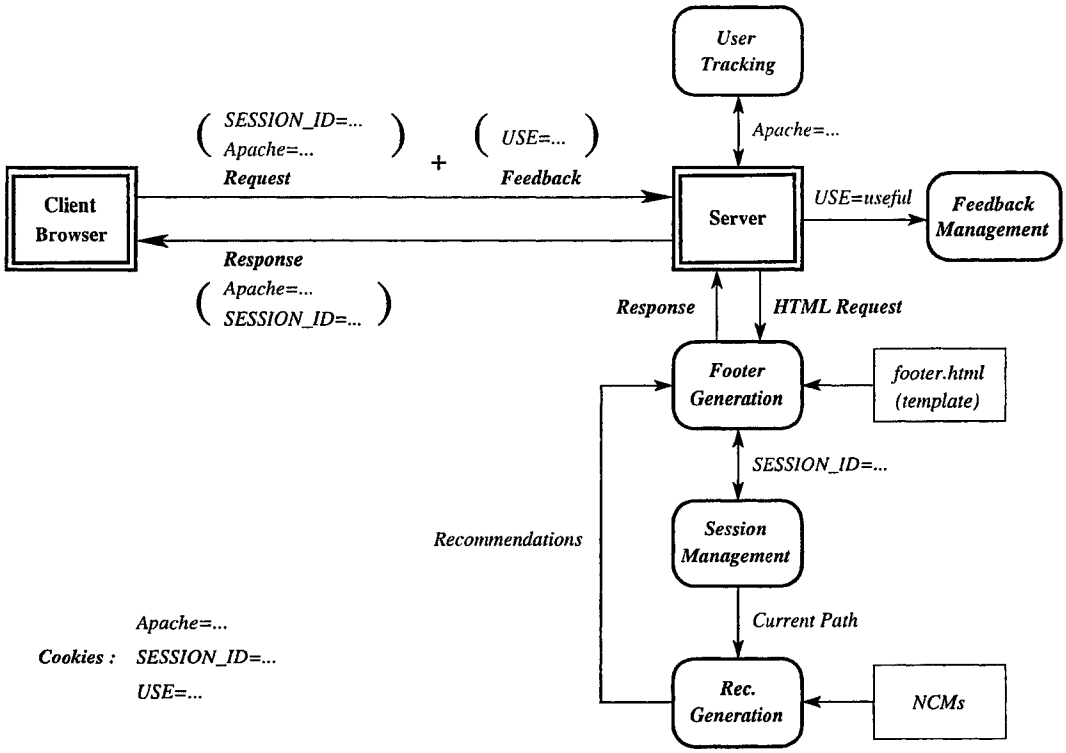


Figure 8.1: NCM-based Recommendation System Architecture

This script module is responsible for recommendation presentation and user feedback gathering in the client browser. The "Footer" is generated in four steps:

(a) A template of the "Footer" (e.g., `footer.html`) is loaded.

(b) The *Session Management* module is called. This module identifies individual sessions using the *SESSION_ID* cookie. If the client request does not carry this cookie, a new session ID will be assigned to the user and saved under this cookie name. The entire session path is saved in a temporary database indexed by the *SESSION_ID* cookie. If the client request does carry this cookie, the carried *SESSION_ID* will be used to retrieve the previous session path to which the new request can be appended.

(c) The *Recommendation Generation* module is called to generate recommendations based on the user's current session path and related NCMs.

(d) The recommendations are embedded into the template to construct the final HTML page to be sent back to client.

4. If the client request carries the *USE* cookie and the value for the cookie is *useful*, then this request is actually a form of user feedback which indicates that this specific document is "useful" to the user. In this case, the *Feedback Management* module is called to handle this information. For example, this feedback can be used to improve the identification of content pages.
5. Finally, the recommendation-embedded HTML page together with the various cookies (*Apache*, *SESSION_ID*) are sent back to client.

8.2.1 User Tracking

In our recommendation system, user tracking is implemented with a cookie named "Apache". Each individual user is assigned a unique user ID when processing the first request from the user. Then this user ID is carried by the "Apache" cookie and sent between client and server to track the specific user. The expiry time of the cookie can be adjusted depending on different preferences. Though this cookie is enabled in our system, it is currently not used for making recommendations.

The purpose of our system is to make personal recommendations based on aggregate user behaviors. Moreover, the learning methods that have been implemented in this system only deal with intra-session information. Therefore what's important here is to identify individual sessions, which has been handled successfully with merely the "SESSION_ID" cookie. However, the "Apache" cookie is still useful for other tasks that require user identification over time, e.g., time series analysis of inter-session user navigation patterns.

8.2.2 Session Management

The identification of user sessions is an important issue in this recommendation system. Basically, user sessions can be identified at the client side or at the server side. Compared to client-side user session identification, at server side, user sessions are more difficult to accurately track. In our system we implemented a server-side session management scheme using a cookie named "SESSION_ID".

Similar to the user tracking, session tracking is done by assigning each individual session a unique session ID when processing the first request of the session. Then this session ID is carried by the "SESSION_ID" cookie and sent between client and server to track the specific session. However, the expiry-period of the "SESSION_ID" cookie is set to last only for the current browser session. Therefore, whenever a session is started, it won't finish until the user terminates the browser. Compared to the session identification approaches used in the experiments, this approach is more accurate because of the additional data used — the "SESSION_ID" cookie. With this new approach, we can know exactly *where* the user finished the session, but still don't know *when*.

For each session ID, the server maintains its corresponding session path in a temporary database. This path is updated continuously with each new request of the session. Finally, this session ID as well as its corresponding path will be removed from the temporary database, if the session path has been unchanged for a certain amount of time (e.g., 24 hours).

8.2.3 Footer Generation

The task of the *Footer Generation* module is to embed the recommendation information into the requested HTML document. This module calls the *Session Management* module to obtain the current session path, and then calls the *Recommendation Generation* module to produce recommendations.

The recommendation information is contained in a JavaScript module appended to the end of the requested HTML document. This script module opens a small pop-up window in the client browser (called "Footer") through which the recommendation system is presented and used. The pop-up window is composed of three sub-windows:

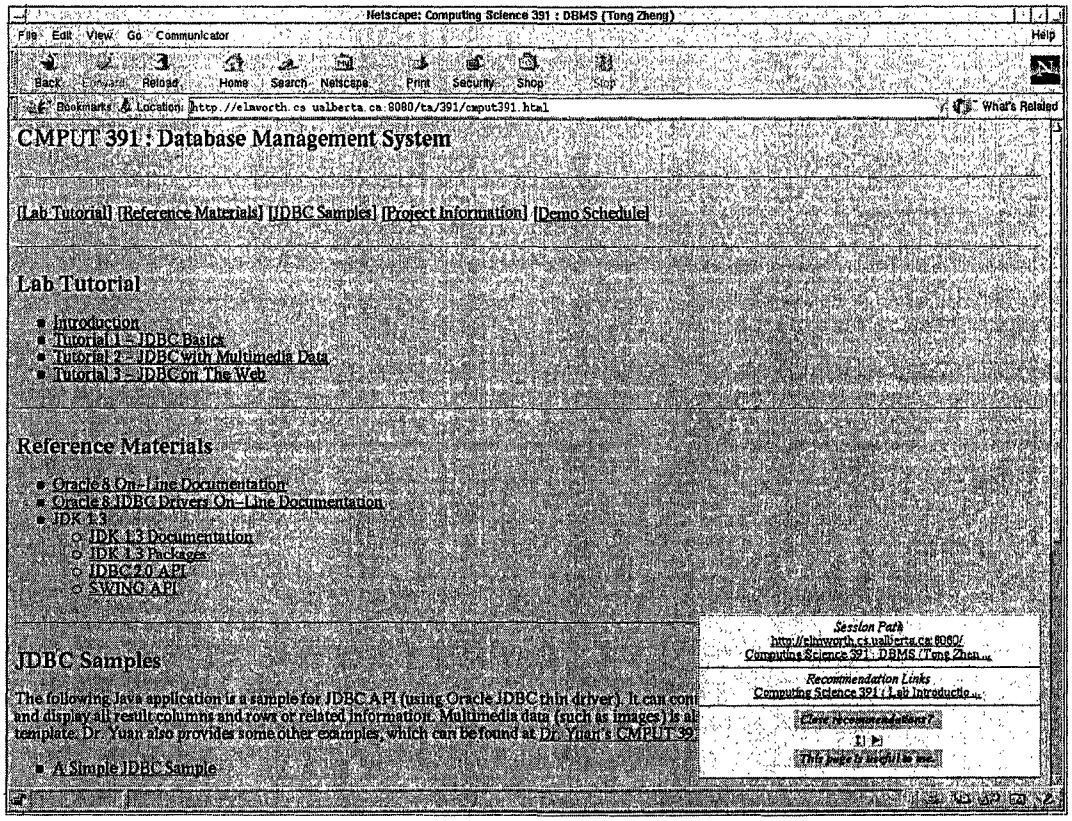


Figure 8.2: Dynamic Recommendation View Example

(1) a "Session Path" window which shows a complete list of all requested documents in the current session (debug use only), (2) a "Recommendation Links" window which presents recommended documents at the current position, and (3) a control window which provides users with abilities to open and close the recommendation window, to change the recommendation window's position in the browser, or to give feedback to the server (see Figure 8.2).

8.2.4 Recommendation Generation

The *Recommendation Generation* module is where we apply the NCMs to generate recommendations. These NCMs were pre-constructed by various learning methods applied to different kinds of web data.

8.2.5 Feedback Management

This recommendation system allows the user to provide feedback to the server in a very simple way. In the control window there is a button labeled "This page is useful to me" (see Figure 8.2). By simply clicking this button the user sends a feedback to the server indicating that this page is a relevant page in the session and therefore can be identified as a content page. The feedback is carried with a cookie named "USE". Through a click of the button a specific request of the current page is sent to the server with "USE=useful". With ordinary requests, "USE=unknown".

8.3 Summary

This NCM-based dynamic recommendation system is only a basic prototype that demonstrates the idea of improving web navigation using knowledge learned from various kinds of web data.

This system requires the user to enable JavaScript and cookies in the client browser. While these two techniques have been widely used across the Internet, some people have concerns about their security issues. Additionally there are also some people who might find them intrusive. On the other hand, the usage data acquisition mechanism used in this system is more accurate and informative than using the ordinary server access logs. Apparently this is the trade-off we have to make between intrusive data gathering and navigation improvement.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

We have developed a framework for web mining, based on a general architecture that decouples data inputs, learning methods, evaluation methods, and visualization. Our initial experiments with our framework focused on improving web navigation, and we developed the idea of navigation compression models (NCMs) to represent the results of learning "better" navigation paths from various kinds of web data (both usage and content data).

The framework tools we have developed, including the architecture for gathering and preprocessing data, applying learning methods, and evaluating the results (both quantitatively and qualitatively via visualization), have provided the basis for initial experiments with navigation improvement as goal.

We have been able to clearly separate heuristics for identifying "relevant" pages (or content pages) from those of integrity and intrusiveness of data gathering, as well as achieving some decoupling of learning and evaluation methods.

In these our first experiments, we have tested various combinations of data selection and preparation methods, learning methods, and evaluation methods. The results of these experiments can be summarized as follows:

- The characteristics of the data determines the potential navigation improvement that can be achieved. For example, a remarkable portion (15~30%, depending on different times of the term and different terms of the year[1]) of the UACS log data records the web usage of UACS faculties and students. These users are familiar with the UACS web site, and in most cases they know exactly what to see and where to go. Such being the case, we can still learn various patterns from their navigation behaviors, but can hardly make any improvement to their navigation. Moreover, the UACS web site is huge and users' interests in this web site are rather diverse, which makes it difficult for both content page identification and learning. In another case, since the users to the "Music Machines" web site are more target-driven and have a much higher revisitation rate, we can expect a much higher navigation improvement on the music machines web log data than on the UACS web log data.

[1]Note that we counted only those requests sent from machines inside the UACS department, and ignored the case that the faculties and students can also access from outside the department. Therefore, the actual number should be larger.

- Our NCM's ability to make recommendations on "relevant" pages is important, but also revealed a number of problems. First, without user feedback obtained through intrusive data gathering, content pages can only be identified heuristically. And the quality of these heuristically identified content pages can only be assessed manually by humans. Second, we used two criteria, support and confidence, to further restrict the content pages obtained with different identification methods. In other words, only those content pages satisfying a minimum support and a minimum confidence are considered true content pages in the process of navigation compression. If we set these criteria too high, many "relevant" pages might be incorrectly identified as auxiliary pages and therefore skipped in the compressed navigation paths. On the other hand, if we set them too low, many "irrelevant" pages could be misidentified as content pages and therefore prevent the corresponding navigation paths to be compressed at a higher rate. How to set these thresholds is still a challenging problem, and currently can only be done empirically.
- Different kinds of NCMs do have different effects on navigation improvement. Our experiments showed that usage-cluster-based NCMs constantly have the best result in navigation improvement measured by NI. Rule-based NCMs tend to have much lower coverage, which greatly limits the navigation improvement that can be achieved. This problem can not be simply solved by generating a very large number of rules, because that will impair the performance of real-time recommendation.
  The result of NI_c is inevitably related to the choice of the recommendation cost (r_cost). Note that compared to cluster-based NCMs, rule-based NCMs are much more cautious in generating recommendations. Therefore, although usage-cluster-based NCMs result in much better NI_c than rule-based NCMs in our experiments, a higher recommendation cost ($r\_cost > 0.05$) will certainly impact this result.
  The problem with content-cluster-based NCMs is that content-based clusters tend to be much larger than usage-based clusters, which means they tend to generate many more recommendations. Therefore, their precision of recommendations will be much lower and their recommendation cost will be much higher.
- We have experimented with different data sets (from the same data source), a variety of heuristics for the identification of users, sessions, and content pages, four different kinds of NCMs, and two different evaluation methods. In these experiments we focused on the intention of the experiments we conducted and the implication of the results we obtained. And we expect the knowledge we have learned from these experiments can be used to guide further experiments with more alternatives on data, learning, and evaluation methods.

In the application of improving web navigation, the knowledge obtained from the learning process can be used in various ways: (a) we can take active moves based on the knowledge, e.g., making dynamic recommendations or creating synthetic index pages; or (b) we can present the knowledge in certain forms (e.g., visualization) to web designers so that they can correspondingly evaluate the web design and manually change it to better assist user navigation. In this research we experimented with the first approach. However, the experimental results are not always encouraging

because we can not guarantee the quality of the knowledge with currently available technologies. Therefore, in some cases the second approach could be a better choice. Such being the case, our web mining system can also be considered as a *web design support system.*

9.2 Future Work

There is much left to do, both in further refining the framework, and in comparing learning and evaluation methods to determine sustainable improvements and sensitivity with respect to data. Our future work can be summarized as follows:

- Our framework can be further refined while more experiments are being conducted with assorted data input, learning and evaluation methods.
- We have demonstrated that we can expect different levels of navigation improvement with different data sets. Therefore, by applying our framework to various kinds of data sets, we can learn more about the sensitivity of this improvement with respect to data.
- The identification of content pages is still a challenging problem. In addition to RL, MFR and MFR-RL, other approaches (e.g., other heuristics or learning models) can be explored for more accurate identification of content pages and for their evaluation.
- In addition to the NCMs experimented in this research, other kinds of NCMs created with different learning methods can also be applied to examine their effects on navigation improvement. Moreover, we hope to investigate the possibility of updating the NCMs incrementally instead of creating NCMs from scratch every time.
- Other evaluation methods as well as other parameter settings for the recommendation mechanism (such as r_cost and r_limit) can also be applied to examine the performance improvement from different standpoints.
- The experiments we have conducted so far used heuristically identified content pages and a simplified user model. We hope to collect some "real" data with user-labeled content pages and recommendation-enabled user access logs, and use that data to validate the various methods and heuristics used in our framework.

Bibliography

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB'94)*, pages 487–499, September 1994.

[2] M. J.A. Berry and G. Linoff. *Data Mining Techniques: For Marketing, Sales, and Customer Support.* John Wiley & Sons, 1997.

[3] K. Bharat and A. Broder. A Technique for Measuring the Relative Size and Overlap of Public Web Search Engines. In *Proceedings of the 7th International World Wide Web Conference*, April 1998.

[4] K. Bharat and A. Broder. Measuring the Web, March 1998. `http://research.compaq.com/SRC/whatsnew/sem.html`.

[5] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Proceedings of the 7th International World Wide Web conference*, April 1998.

[6] C. Bron and J. Kerbosch. Algorithm 457: Finding All Cliques of an Undirected Graph. *Communications of the ACM*, 16(9):575–577, 1973.

[7] L.D. Catledge and J.E. Pitkow. Characterizing Browsing Strategies in the World-Wide Web. *Computer Networks and ISDN Systems*, 27(6):1065–1073, 1995.

[8] S. Chakrabarti, M. Berg, and B. Dom. Focused Crawling: a New Approach to Topic-specific Web Resource Discovery. In *Proceedings of the 8th World Wide Web conference*, 1999.

[9] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Scalable Feature Selection, Classification and Signature Generation for Organizing Large Text Databases into Hierarchical Topic Taxonomies. *VLDB Journal*, 7(3):163–178, 1998.

[10] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Mining the Link Structure of the World Wide Web. *IEEE Computer*, 1999.

[11] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text. In *Proceedings of the 7th World Wide Web conference*, 1998.

[12] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced Hypertext Categorization Using Hyperlinks. In *Proceedings of SIGMOD'98, ACM International Conference on Management of Data*, pages 307–318, Seattle, US, 1998. ACM Press, New York, US.

[13] J. Chambers, W. Cleveland, B. Kleiner, and P. Tukey. *Graphical Methods for Data Analysis*. Wadsworth Int'l Group, Belmont, CA, 1983.

[14] S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record*, 26(1):65–74, March 1997.

[15] M. S. Chen, J. S. Park, and P. S. Yu. Data mining for Path Traversal Patterns in a Web Environment. In *Sixteenth International Conference on Distributed Computing Systems*, pages 385–392, 1996.

[16] E.H. Chi, J. Pitkow, J. Mackinlay, P. Pirolli, and J. Konstan. Visualizing the Evolution of Web Ecologies. In *Proceedings of CHI'98*, 1998.

[17] J. Cho, H. Garcia-Molina, and L. Page. Efficient Crawling Through URL Ordering. In *Proceedings of the 7th International World Wide Web conference*, April 1998.

[18] A. Cockburn and B. Mckenzie. What Do Users Do? An Empirical Analysis of Web Use. *International Journal of Human-Computer Studies*, 54(6):903–922, 2001.

[19] R. Cooley, B. Mobasher, and J. Srivastava. Grouping Web Page References into Transactions for Mining World Wide Web Browsing Patterns. In *Proceedings of the 1997 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'97)*, November 1997.

[20] R. Cooley, B. Mobasher, and J. Srivastava. Web Mining: Information and Pattern Discovery on the World Wide Web. In *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, November 1997.

[21] R. Cooley, B. Mobasher, and J. Srivastava. Data Preparation for Mining World Wide Web Browsing Patterns. *Journal of Knowledge and Information Systems*, 1(1), 1999.

[22] D.R. Cutting, D. Karger, J. Pedersen, and J.W. Tukey. Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections. In *Proceedings of SIGIR-92*, pages 318–329, Copenhagen, Denmark, 1992.

[23] G.Z. Dong and J.Y. Li. Interestingness of Discovered Association Rules in Terms of Neighborhood-Based Unexpectedness. In *Proceedings of the 2nd Pacific-Asia Confcwerence on Knowledge Discovery and Data Mining (PAKDD'98)*, pages 72–86, Melbourne, Australia, April 1998.

[24] R. Dugad and U.B. Desai. A Tutorial on Hidden Markov Models. Technical Report SPANN-96.1, Signal Processing and Artificial Neural Networks Laboratory, Department of Electrical Engineering, Indian Institute of Technology — Bombay, Powai, Mumbai 400 076, India, May 1996.

[25] B. Eberman, B. Fidler, R.A. Iannucci, C. Joerg, L. Kontothanassis, D.E. Kovalcin, P. Moreno, M.J. Swain, and JM.V. Thong. Indexing Multimedia for the Internet. Technical report, Cambridge Research Laboratory, March 1999.

[26] M. Frauenfelder. The Future of Search Engines, September 1998. `http://websearch.about.com/library/weekly/aa082101b.htm`.

[27] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring Web Communities from Link Topologies. In *Proceedings of The Ninth ACM Conference on Hypertext and Hypermedia*, 1998.

[28] L.F. Gu, J.Y. Li, and H.X. He. Association Rule Discovery with Unbalanced Class Distribution. In *Proceedings of the 16th Australian Joint Conference on Artificial Intelligence (AI'2003)*, Perth, Australia, December 2003.

[29] E.H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering Based On Association Rule Hypergraphs. In *Proceedings of the SIGMOD'97 Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 9–13, 1997.

[30] E.H. Han, G. Karypis, V. Kumar, and B. Mobasher. Hypergraph Based Clustering in High-Dimensional Data Sets: A Summary of Results. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 21(1):15–22, 1998.

[31] J.W. Han, J. Chiang, and S. Chee. DBMiner: A System for Data Mining in Relational Databases and Data Warehouses. In *CASCON'97: Meeting of Minds*, pages 249–260, Toronto, Canada, November 1997.

[32] M. Hersovici, M. Jacovi, Y.S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur. The Shark-search Algorithm - An Application: Tailored Web Site Mapping. In *Proceedings of the 7th International World Wide Web Conference*, 1998.

[33] R. Hightower, L.T. Ring, J.I. Helfman, B. Bederson, and J.D. Hollan. Graphical Multiscale Web Histories: A Study of PadPrints. In *Proceedings of ACM Hypertext 98 Conference*, pages 58–65, 1998.

[34] B. Kahle. Archiving the Internet. *Scientific American*, March 1997.

[35] P. Kahn and K. Lenk. *Mapping Websites: Digital Media Design.* RotoVision SA, 2001.

[36] G. Karypis, E.H. Han, and V. Kumar. Chameleon: A Hierarchical Clustering Algorithm Using Dynamic Modeling. *IEEE Computer: Special Issue on Data Analysis and Mining*, 32(8):68–75, 1999.

[37] J. Kleinberg, S.R. Kumar, P. Raghavan, S. Rajagopalan, and A.S. Tomkins. The Web as a Graph: Measurements, Models and Methods. In *Proceedings of the International Conference on Combinatorics and Computing*, 1999.

[38] J.M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. In *Proceedings of the 9th ACMSIAM Symposium on Discrete Algorithms*, 1998.

[39] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo. Finding Interesting Rules from Large Sets of Discovered Association Rules. In *Third International Conference on Information and Knowledge Management (CIKM'94)*, pages 401–407, November 1994.

[40] B. Lavoie and H.F. Nielsen. Web Characterization Terminology & Definitions Sheet, May 1999. `http://www.w3.org/1999/05/WCA-terms/`.

[41] E. Lim and J. Paynter. Design Considerations for Web Site Navigation. In *UniForum NZ Conference*, 1998.

[42] B. Liu, W. Hsu, S. Chen, and Y.M. Ma. Analyzing the Subjective Interestingness of Association Rules. *IEEE Intelligent Systems*, 15(5):47–55, 2000.

[43] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Building Domain-Specific Search Engines with Machine Learning Techniques. In *Proceedings of the AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace*, 1999.

[44] A.K. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 3(2):127–163, 2000.

[45] J. McQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the 5th Berkeley Symponium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[46] T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[47] B. Mobasher, R. Cooley, and J. Srivastava. Automatic Personalization Based on Web Usage Mining. Technical Report TR99-010, Department of Computer Science, Depaul University, 1999.

[48] B.H. Murray and A. Moore. Sizing the Internet. A white paper, Cyveillance, July 2000.

[49] Y.H. Niu. Web Knowledge and Information Visualization (WebKIV). Master's thesis, University of Alberta, May 2003.

[50] Y.H. Niu, T. Zheng, R. Goebel, and J.Y. Chen. WebKIV: Visualizing Structure and Navigation for Web Mining Applications. In *2003 IEEE/WIC International Conference on Web Intelligence (WI'03)*, Halifax, Canada, October 13-17 2003.

[51] S. Oyanagi, K. Kubota, and A. Nakase. Application of Matrix Clustering to Web Log Analysis and Access Prediction. In *WebKDD 2001 Workshop in conjunction with the ACM-SIGKDD 2001*, pages 13–21, San Francisco, CA, USA, August 2001.

[52] G. Pagallo and D. Haussler. Boolean Feature Discovery in Empirical Learning. *Machine Learning*, 5:71–100, 1990.

[53] P. Pantel and D. Lin. Document clustering with committees. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 199–206, Tampere, Finland, August 2002.

[54] J.S. Park, M.S. Chen, and P.S. Yu. An Efficient Hash-Based Algorithm for Mining Association Rules. In *Proceedings of ACM SIGMOD*, pages 175–186, May 1995.

[55] J.S. Park, M.S. Chen, and P.S. Yu. Using a Hash-Based Method with Transaction Trimming and Database Scan Reduction for Mining Association Rules. *IEEE Trans. on Knowledge and Data Engineering*, 9(5):813–825, October 1997.

[56] M. Perkowitz and O. Etzioni. Adaptive Web Sites: Automatically Synthesizing Web Pages. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[57] M. Perkowitz and O. Etzioni. Adaptive Web Sites: Conceptual Cluster Mining. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1999.

[58] G. Piatesky-Shapiro. Discovery, Analysis and Presentation of Strong Rules. In *Knowledge Discovery in Databases*, pages 229–248, Cambridge, MA, USA, 1991. MIT Press.

[59] P. Pirolli, J. Pitkow, and R. Rao. Silk from a Sow's Ear: Extracting Usable Structures from the Web. In *Proceedings of the 1996 Conference on Human Factors in Computing Systems (CHI-96)*, Vancouver, Canada, 1996.

[60] J.E. Pitkow and M.M. Recker. Results from the First World-Wide Web User Survey. *Computer Networks and ISDN Systems*, 27(2):243–254, 1994.

[61] L.R. Rabiner and B.H. Juang. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine*, pages 4–16, January 1986.

[62] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[63] C.J. Van Rijsbergen. *Information Retrieval.* Butterworths, London, second edition, 1979.

[64] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval.* Computer Science Series. McGraw-Hill, New York, 1983.

[65] S. Schechter, M. Krishnan, and M.D. Smith. Using Path Profiles to Predict HTTP Requests. In *Proceedings of the Seventh International World Wide Web Conference*, Brisbane, Australia, April 1998.

[66] C. Shahabi, F.B. Kashani, and J. Faruque. A Reliable, Efficient, and Scalable System for Web Usage Data Acquisition. In *WebKDD 2001 Workshop in conjunction with the ACM-SIGKDD 2001*, pages 71–79, San Francisco, CA, USA, August 26 2001.

[67] C. Shahabi, F.B. Kashani, J. Faruque, and A. Faisal. Feature Matrices: A Model for Efficient and Anonymous Mining of Web Navigations. Technical

Report USC-CS-00-736, Computer Science Department, University of Southern California.

[68] C. Shahabi, F.B. Kashani, J. Faruque, and A. Faisal. Feature Matrices: A Model for Efficient and Anonymous Web Usage Mining. In *EC-Web 2001*, pages 280–294, Munich, Germany, September 2001.

[69] B.D. Sheth. A Learning Approach to Personalized Information Filtering. Master's thesis, Massachusetts Institute of Technology, February 1994.

[70] A. Silberschatz and A. Tuzhilin. On Subjective Measures of Interestingness in Knowledge Discovery. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 275–281, Montreal, Canada, August 1995.

[71] J. Srivastava, R. Cooley, M. Deshpande, and P.N. Tan. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. In *SIGKDD Explorations*, volume 1, Issue 2, January 2000.

[72] R. Stout. *Web Site Stats: Tracking Hits and Analyzing Traffic.* McGraw-Hill, 1997.

[73] M.J. Swain. Searching for Multimedia on the World Wide Web. Technical report, Cambridge Research Laboratory, March 1999.

[74] L. Tauscher and S. Greenberg. How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems. *International Journal of Human Computer Studies, Special issue on World Wide Web Usability*, 47(1):97–137, 1997.

[75] L. Tauscher and S. Greenberg. Revisitation Patterns in World Wide Web Navigation. In *Proceedings of the ACM SIGCHI'97 Conference on Human Factors in Computing Systems*, Atlanda, Georgia, USA, March 1997. ACM Press.

[76] W3C. Extensible Markup Language (XML). `http://www.w3.org/XML/`.

[77] W3C. HyperText Markup Language (HTML). `http://www.w3.org/MarkUp/`.

[78] R. Weiss, B. Velez, M.A. Sheldon, C. Namprempre, P. Szilagyi, A. Duda, and D.K. Gifford. HyPursuit: A Hierarchical Network Search Engine that Exploits Content-Link Hypertext Clustering. In *Proceedings of the Seventh ACM Conference on Hypertext*, pages 180–193, 1996.

[79] A. Wexelblat and P. Maes. Footprints: History-Rich Tools for Information Foraging. In *Proceedings of CHI'99*, pages 270–277, May 1999.

[80] O.R. Zaïane, J.W. Han, Z.N. Li, and J. Hou. Mining Multimedia Data. In *CASCON'98: Meeting of Minds*, pages 83–96, Toronto, Canada, November 1998.

[81] O.R. Zaïane, M. Xin, and J.W. Han. Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. In *Advances in Digital Libraries (ADL'98)*, pages 19–29, Santa Barbara, Canada, April 1998.

[82] T.S. Zhu, R. Greiner, and G. Häubl. Important Page Predicting for Internet Recommendation: A Machine Learning Way. In *International Conference on Fuzzy Systems and Knowledge Discovery*, Orchid Country Club, Singapore, November 2002.

[83] T.S. Zhu, R. Greiner, and G. Häubl. An Effective Complete-Web Recommender System. In *The 12th International World Wide Web Conference (WWW2003)*, Budapest, HUNGARY, May 2003.

[84] T.S. Zhu, R. Greiner, and G. Häubl. Learning a Model of a Web User's Interests. In *The 9th International Conference on User Modeling (UM2003)*, Johnstown, USA, June 2003.

[85] I. Zukerman, D.W. Albrecht, and A.E. Nicholson. Predicting Users' Requests on the WWW. In *User Modeling: Proceedings of the Seventh International Conference (UM-99)*, pages 275–284, June 1999.