Web usage mining for web personalization

Zhou, Baoyao

2007

Zhou, B. Y. (2007). Web usage mining for web personalization. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/2586>

<https://doi.org/10.32657/10356/2586>

Nanyang Technological University

Downloaded on 26 Apr 2023 22:12:55 SGT

NANYANG TECHNOLOGICAL UNIVERSITY

Web Usage Mining for Web Personalization

BAOYAO ZHOU

A thesis submitted to the
Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

School of Computer Engineering

2006

Abstract

With the explosive growth of information available on the World Wide Web, it has become much more difficult to access relevant information from the Web. Various web services have been developed to assist users' web browsing activities. However, the existing web services provided are far from satisfactory to support the needs of different users. One possible solution to solve this problem is web personalization, which aims to customize the content and structure of a website to the needs of specific users by taking advantage of the knowledge acquired from the analysis of users' access behaviors. Discovering and understanding users' web access behaviors, especially surfing interests and habits, are very important for supporting effective web personalization. Web usage mining is a promising approach for discovering users' web access patterns from web usage logs. The objective of this research is to investigate novel web usage mining approaches for discovering users' web access patterns and effective techniques for web personalization.

This research aims to investigate techniques for enhancing web usage mining in both access pattern representation and mining algorithms, and techniques for applying the discovered access patterns to support effective personalized services on the Web and Semantic Web. The contributions of this research are listed as follows:

- An efficient sequential pattern mining algorithm, called CSB-mine (Conditional Sequence Base mining algorithm), is proposed for discovering frequent sequential patterns efficiently from sequence databases, especially the web usage logs.
- A web recommender system, named *SWARS* (Sequential Web Access based Recommender System), is developed for matching a user's current access sequence efficiently and recommending related web pages to the user effectively based on the sequential access patterns mined by the proposed CSB-mine algorithm.
- A web usage mining approach is proposed for discovering a specific kind of periodic association access patterns of individual users from web usage logs. The proposed approach incorporates fuzzy set theory into Formal Concept Analysis (FCA) for constructing a

novel user behavior model, called Personal Web Usage Lattice, from which periodic association access patterns of the user can be extracted and visualized.

- A Personal Web Usage Lattice based approach is proposed for periodic web personalization. Different from non-periodic approaches, the proposed periodic web personalization approach can determine efficiently which resources a user is most probably interested in during a given time period based on the Personal Web Usage Lattice of the user, without the use of the user's current access information. This makes it possible to perform more costly personalized resource preparation in advance rather than in real-time for periodic web personalization.
- A semantic web personalization framework is proposed. In the proposed framework, Web Usage Ontologies are generated automatically based on the proposed Web Usage Lattice models. The periodic web personalization approach is extended for supporting periodic-based personalized services on the Semantic Web.

Acknowledgement

I would like to express my sincere thanks and appreciation to my supervisor, Assoc. Prof. Hui Siu Cheung, for his conscientious guidance and thoughtful criticism. Without his constant encouragement and invaluable suggestions, this thesis would not have been completed. He has also always kept on pushing me to advance to higher research levels.

I would like to thank Dr. Alvis Fong Cheuk Ming and Dr. Chang Kuiyu for their review and invaluable comments on my research.

I am also thankful to Nanyang Technological University (NTU) and Singapore Millennium Foundation Ltd (SMF) for providing me the financial support to do my research in Singapore.

My gratitude also goes to Mr. Teo Choo Eng and Ms. Eng Hui Fang, the laboratory technicians of the Database Technology Laboratory, for their friendly supports and help.

I appreciate to my parents, brother and sister-in-law, for their love and support from thousands of miles away. In particular, I would like to express my deepest gratitude to my wife, Iris Huang Jiahong, for her constant supports and care. I would not have had these days without her endless encouragement.

Last but not least, I want to thank all my friends and colleagues, who assisted me in many ways throughout the duration of the research.

Contents

Abstract	i
Acknowledgement	iii
1 Introduction	1
1.1 Web Mining	1
1.2 Web Usage Mining	2
1.3 Web Personalization and Recommendation	5
1.4 Semantic Web Mining	6
1.5 Objectives	7
1.6 Major Contributions	8
1.7 Organization of the Thesis	9
2 Web Usage Mining	11
2.1 Web Usage Data	11
2.2 Preprocessing	12
2.2.1 Data Cleaning	12
2.2.2 User Identification	13
2.2.3 Session Identification	13
2.2.4 Other Preprocessing Tasks	14
2.3 Pattern Discovery Techniques	14
2.3.1 Statistical Approaches	14
2.3.2 Association Rule Mining	14
2.3.3 Sequential Pattern Mining	17
2.3.4 Clustering	20
2.3.5 Classification	22
2.4 Summary	23

3	Efficient Sequential Access Pattern Mining	25
3.1	Problem Statement	27
3.2	The WAP-tree and the WAP-mine Algorithm	28
3.3	The CSB-mine Algorithm	30
3.3.1	Preprocessing	31
3.3.2	Constructing Event Queues for Conditional Sequence Base	31
3.3.3	Constructing Sub-Conditional Sequence Base	33
3.3.4	Single Sequence Testing for Conditional Sequence Base	35
3.3.5	The Complete CSB-mine Algorithm	36
3.4	Performance Evaluation	38
3.4.1	Datasets	38
3.4.2	Experiments	38
3.5	Summary	40
4	Sequential Web Access Based Recommender System	41
4.1	Web Personalization and Recommendation	42
4.1.1	Rule-based Filtering Approaches	42
4.1.2	Content-based Filtering Approaches	42
4.1.3	Collaborative Filtering Approaches	43
4.1.4	Web Usage Mining Approaches	43
4.2	System Architecture	44
4.3	Pattern-tree Construction	45
4.4	Recommendation Rules Generation	48
4.5	Performance Evaluation	50
4.5.1	Evaluation Measures	51
4.5.2	Experiments	52
4.6	Summary	54
5	Periodic Association Access Pattern Mining	56
5.1	Periodic Pattern Mining Approaches	57
5.1.1	Mining Periodic Association Rules	58
5.1.2	Mining Periodic Sequential Patterns	59
5.2	Fuzzy Set Theory	59
5.3	Formal Concept Analysis	61

5.3.1	Formal Context and Concept Lattice	61
5.3.2	Applications of FCA	62
5.4	Discovering Periodic Web Access Patterns	64
5.4.1	Preprocessing	64
5.4.2	Constructing Personal Web Usage Lattice	67
5.4.3	Extracting Periodic Association Access Patterns	73
5.5	Visualizing Periodic Association Access Patterns	76
5.6	Summary	79
6	Periodic Web Personalization	81
6.1	System Architecture	82
6.2	Personalized Resources Generation	83
6.3	Performance Evaluation	89
6.3.1	Evaluation Measures	89
6.3.2	Experiments	90
6.4	Summary	94
7	Periodic-based Semantic Web Personalization	96
7.1	The Semantic Web	97
7.2	Web Usage Mining for the Semantic Web	99
7.3	Ontology Generation	100
7.4	Web Services and Semantic Web Services	101
7.5	System Overview	103
7.6	Automatic Web Usage Ontology Generation	104
7.6.1	Global Web Usage Lattice Construction	104
7.6.2	Personal Web Usage Lattice Construction	107
7.6.3	Web Usage Ontology Generation	108
7.7	Personalized Resources Generation Service	115
7.8	Service Specification Creation	115
7.8.1	Creating the Service Model	116
7.8.2	Creating the Service Grounding	117
7.8.3	Creating the Service Profile	119
7.9	Prototype Service Requester System	120
7.10	Summary	120

8	Conclusions	122
8.1	Summary	122
8.2	Future Work	124
8.2.1	Further Enhancement of the Proposed Research Works	124
8.2.2	Web User Clustering	125
8.2.3	Client-side Access Logs Mining for User Access Behavior	126
8.2.4	FCA based Web Usage Retrieval	128
8.2.5	Extending the Semantic Web Personalization Approach	130
8.2.6	Supporting Other Semantic Web Personalization Services	131
A	List of Publications	133
A.1	International Journal Papers	133
A.2	International Conference / Workshop Papers	133
	Bibliography	135

List of Tables

2.1	Useful statistical information discovered from web logs.	15
2.2	Pattern discovery techniques for web usage mining.	24
3.1	A sample database of web access sequences.	27
3.2	The sequential access patterns of the sample database.	37
4.1	Performance results of the proposed recommender system.	53
5.1	A cross table of a formal context.	62
5.2	An example of a semantically enriched web usage log.	65
5.3	A cross table of an example fuzzy periodic Web Usage Context of a user. . .	69
5.4	A cross table of the formal context converted from the example Web Usage Context given in Table 5.3.	72
5.5	Periodic access patterns.	75
6.1	Experimental datasets.	91
7.1	An example of a semantic web usage log.	105

List of Figures

1.1	A fragment of a web server log in the W3C extended log file format.	3
1.2	Processes of web usage mining.	3
1.3	The proposed research framework.	8
2.1	A fragment of a web server log from “North Latitude One BBS”.	12
3.1	The WAP-tree for frequent sub-sequences in Table 3.1.	29
3.2	Conditional WAP-trees for WAP-tree in Figure 3.1.	30
3.3	Overview of the proposed CSB-mine algorithm.	31
3.4	<i>Init-CSB</i> with the Header Table and event queues.	33
3.5	Construction of $CSB(a)$ based on <i>Init-CSB</i>	34
3.6	Performance results of CSB-mine vs. WAP-mine vs. PrefixSpan based on different support thresholds.	40
3.7	Performance results of CSB-mine vs. WAP-mine vs. PrefixSpan based on different numbers of sequences.	40
4.1	The architecture of the proposed web recommender system.	45
4.2	An example browser display of the requested web page with recommended links.	46
4.3	The Pattern-tree derived from the sequential access patterns given in Table 3.2.	47
4.4	Performance results of the proposed recommender system based on different numbers of recommended resources.	54
4.5	Performance results of the proposed recommender system based on m-step satisfaction.	54
5.1	The concept lattice of the formal context given in Table 5.1.	63
5.2	The proposed approach.	64
5.3	Member function $\mu_p(t, m_p)$	68
5.4	Member function $\mu_r(g, m_r)$	69

5.5	Personal Web Usage Lattice.	71
5.6	The concept lattice of the formal context given in Table 5.4.	73
5.7	Converting fuzzy set of periodic attributes into time intervals.	77
5.8	Converting fuzzy membership values of resources into meaningful interest levels.	78
5.9	Converting fuzzy support and confidence into frequency and quality levels.	78
5.10	Visualizing periodic association access patterns.	80
6.1	The architecture of the proposed periodic web personalization system.	83
6.2	The searching process for period-supported activities.	88
6.3	Applicability of the overall web personalization based on the different durations of period conditions.	92
6.4	Satisfaction of the overall web personalization for predefined period conditions based on the number of personalized resources.	93
6.5	Performance comparison based on the applicability measure.	94
6.6	Performance comparison based on the satisfaction measure.	95
7.1	Architecture of Web Services.	101
7.2	Semantic Web personalized services.	103
7.3	The proposed approach for the automatic generation of Personal Web Usage Ontology.	104
7.4	An example of the Global Web Usage Lattice.	107
7.5	Representing an activity class in OWL.	112
7.6	Representing an activity instance in OWL.	114
7.7	Inputs and outputs of the PersonalizedResourcesGeneration process.	116
7.8	Prototype Service Requester System for personalized resources generation ser- vice.	121

Chapter 1

Introduction

The World Wide Web serves as a huge, widely distributed, global information service center for news, advertisements, consumer information, financial management, education, government, e-commerce and many other information services. Web surfing has become an important daily activity for many people. However, with the explosive growth of information available on the World Wide Web, it has become much more difficult to access relevant information from the Web. Although various techniques have been developed to assist users' web browsing activities, the existing web services provided are far from satisfactory to support the needs of different users, especially with the emergence of the Semantic Web [BLHL01]. Therefore, further research needs to be carried out to develop efficient and effective techniques and services for web users. To achieve this, web mining is a promising direction to provide enhanced web services, especially on the Semantic Web.

1.1 Web Mining

Besides a large amount of content information, web pages also contain a rich and dynamic collection of hyperlink information. Web page access and usage information are recorded in web logs. Web mining [KB00, HC02, Cha02, KJ04] is the use of data mining techniques to automatically discover and extract useful information from web page documents and services. Web mining research integrates research from several communities including databases, information retrieval, machine learning and natural language processing. Web mining tasks are mainly divided into three categories, namely web content mining, web structure mining, and web usage mining.

Web content mining [LCN03, ABA04, JdR05] aims to discover useful information from

web content or documents. Basically, web content contains textual data (plain text and anchor text), image, audio, video and metadata. Most of the web content data are either unstructured (i.e., free text) or semi-structured (i.e., HTML documents). The goals of web content mining include supporting information finding (e.g., search engines), filtering information based on user profiles, modeling data on the Web, and integrating web data for more sophisticated queries. Text mining [Tan99] and multimedia data mining [ZHL⁺98] techniques can be used for mining web content data.

Web structure mining [ZCS⁺01, FS04, LK05] aims to discover the link structure model based on the topology of hyperlinks on the Web. Web structure includes both inter-structure and intra-structure. Inter-structure refers to the structure formed by hyperlinks between different web pages, while intra-structure refers to the structure within an individual web document. The link structure model can be used for categorizing web pages and computing the similarity measures or relationships between web pages. It is also useful for discovering authoritative web pages, the structure of web pages itself, and the nature of the hierarchy of hyperlinks in a website of a particular domain.

Web usage mining [SCDT00, WAS05, FL05], also known as web log mining, aims to discover interesting usage patterns and knowledge from web browsing data that are stored in web logs. Web usage mining can also be used for learning user profiles and navigation patterns. The acquired knowledge can then be used for applications such as personalization [EV03, BFR⁺03, PPPS03], system improvement [YZL01, SOBB03], site modification [Spi00, NSER02], business intelligence [Abr03] and usage characterization [PP99].

In this research, we focus on investigating advanced web usage mining techniques to provide enhanced web services.

1.2 Web Usage Mining

Web usage mining [SCDT00, WAS05, FL05] performs mining on web usage data, or web logs, which record access requests from users to one or multiple websites as a sequence of requested URLs with timestamps. Web usage data are sometimes referred to as *clickstream* data because each entry corresponds to a mouse click in a client-side browser. It mainly includes web server logs, proxy server logs and client browser logs. A fragment of a web server log in the W3C extended log file format is given in Figure 1.1. In addition, it can also include data from user profiles, registration data, cookies, user queries, bookmarks and any other data derived from the interactions of users while surfing on the Web.

```

#Software: Microsoft Internet Information Server 4.0
#Version: 1.0
#Date: 2001-01-15 00:09:34
#Fields: time c-ip cs-method cs-uri-stem sc-status
2001-01-15 00:09:33 202.201.32.114 GET /default.htm 200
2001-01-15 00:09:33 202.201.32.114 GET /images/first_123.gif 200
2001-01-15 00:09:33 202.201.32.114 GET /images/first_71.GIF 200
2001-01-15 00:10:04 202.201.32.114 GET /images/first_51.GIF 200
2001-01-15 00:10:04 202.201.32.114 GET /images/first_124.gif 200
2001-01-15 00:22:38 209.188.67.130 GET /frame/english.htm 404
2001-01-15 00:22:52 209.188.67.130 GET /frame/eng01.htm 404
2001-01-15 00:43:28 166.111.231.24 GET /default.htm 200
2001-01-15 00:43:28 166.111.231.24 GET /first.html 200
2001-01-15 00:43:28 166.111.231.24 GET /images/first_01.gif 200
2001-01-15 00:43:28 166.111.231.24 GET /images/zyzl.jpg 304
2001-01-15 00:43:28 166.111.231.24 GET /images/first_02.gif 200

```

Figure 1.1: A fragment of a web server log in the W3C extended log file format.

Figure 1.2: Processes of web usage mining.

Web usage mining consists of three processes, namely preprocessing, pattern discovery, and pattern analysis as shown in Figure 1.2.

- *Preprocessing.* It converts the original web usage logs into data abstractions necessary for pattern discovery. The main tasks include data cleaning, user identification, session identification and other further preprocessing.
- *Pattern Discovery.* It adopts various data mining techniques for discovering access patterns from the preprocessed web usage logs. The main techniques include statistical analysis, association rule mining, sequential pattern mining, clustering and classification.
- *Pattern Analysis.* Once the access patterns have been identified, they are then required to be analyzed to determine how that information can be used. Pattern analysis filters out uninteresting patterns from the set of patterns found during the pattern discovery process. The exact analysis methodology is usually governed by the purpose of the application.

Web usage mining techniques have been applied to many practical applications [SCDT00] including the followings:

- *Personalization* [EV03, BFR⁺03, PPPS03]. Web personalization is the process of customizing the content and structure of a website to the needs of specific users. The website can be personalized through highlighting the existing hyperlinks, dynamically inserting new hyperlinks that seem to be interesting to the current user, or even creating new index pages.
- *System Improvement* [YZL01, SOBB03]. By analyzing the web traffic behavior, the frequent access patterns can be discovered and applied for developing the policies of web caching, document pre-fetching and data distribution. As such, the performance of the system can be improved. Web usage mining is also useful for detecting intrusions and frauds by discovering frequent unexpected access patterns and outliers.
- *Site Modification* [Spi00, NSER02]. The ways the users accessing a website are restricted by the website's link structure. The organization of web pages within the website has a great influence on the quality of the web services provided. Web usage mining can be applied to provide the insight on the organization of the website in order to improve user browsing activities.
- *Business Intelligence* [Abr03]. Web logs of e-commerce websites can provide information on how customers purchase products online. Web usage mining can be used to gather business intelligence and identify potential customers to improve sales and advertisements.
- *Usage Characterization* [PP99]. Web usage characterization analyzes the overall patterns of web usage, including server access patterns, the kind of data accessed, bytes transferred, the popularity of resources, etc. This will enable us to look at the dynamics of the Web and how it is growing. Note that usage characterization concerns only on the general patterns of web usage, rather than specific users or websites.

With the increasing demand on the quality of web services, especially for e-commerce, it has become much more important for analyzing web usage data to gain a better understanding on user access behavior and applying the knowledge to provide better services to users. Web usage mining is a powerful technique for achieving this objective.

Although various techniques have been proposed to support web usage mining, many issues still need to be tackled in order to provide high quality web services. These issues [SCDT00, HC02, ZC02, KJ04] are summarized and listed as follows:

- *Improving the efficiency and scalability of mining algorithms.* Web usage mining is

the application of data mining techniques to discover knowledge from web usage data. However, efficiency and scalability are still the key issues [HK05] in the implementation of data mining applications. Many existing data mining algorithms work well in small datasets. But, they are unable to deal with a huge amount of data efficiently. Usually, web usage data are too huge and complex for efficient data mining. Therefore, some existing algorithms need to be improved or new scalable algorithms need to be developed to mine large amount of web usage data.

- *Discovering high quality knowledge.* The quality of the discovered knowledge influences directly on the quality of the web services provided. Although various access patterns can be discovered by existing web usage mining techniques, we still need to discover higher quality access patterns to support better web services. In order to discover high quality knowledge, new data mining methods and techniques are required.
- *Applying the discovered knowledge for advanced and practical web applications.* Once access patterns have been discovered, they should further be analyzed and applied to advanced and practical web applications effectively. However, many existing web usage mining approaches only focus on discovering access patterns rather than utilizing them practically. The interesting access patterns should be visualized and interpreted to users. Also, efficient pattern matching algorithms are required to support practical web services.
- *Discovering semantic information.* Since web logs lack of semantic information about web pages visited by users, it is difficult to understand the preferences and intentions of users. Therefore, the semantics in web content should be used for improving the results of existing web usage mining approaches. With the development of the Semantic Web [BLHL01, BHS02, SBH02], web usage mining techniques need to be enhanced using Semantic Web technologies.

1.3 Web Personalization and Recommendation

The content and structure of the Web are too huge and complicated, such that users are sometimes unable to access their interested information while surfing on the Web. The usability of a website can be further improved by personalizing it for specific users according to their intended access behaviors. Web personalization [EV03, BFR⁺03, PPS03] is defined as the process of adapting the content and structure of a website to the needs of specific users by taking advantage of the knowledge acquired from the analysis of users' access behaviors.

The objective of a web personalization system is to provide users with information they need without asking them explicitly.

In general, web personalization includes the following major tasks:

- *Data collection.* Useful data for web personalization including web content, web structure, web usage and user profiles are collected.
- *Data analysis.* Useful knowledge is discovered from the collected data. The main approaches include rule-based filtering (such as Amazon.com and Dell.com), content-based filtering [Lie95], collaborative filtering [KMM⁺97, JFM97, SKCM01] and web usage mining approaches [MDLN01, LAR02, MGPL04, MDLN02, GH03, ZHC04, Mob99]. In this research, we focus on usage-based personalization.
- *Website publishing.* A publishing mechanism is used in order to present web resources in a uniform manner to the end users. The website can be personalized through highlighting existing hyperlinks, dynamically inserting new hyperlinks that are predicted to be interesting to the current user, or even creating new index pages, such as Google Personalized Web Search [Goo] and Adaptive Web Sites [PE00].

Recommendation of web resources that are related to the interests and preferences of users is one of the most commonly offered web personalization functions. Such recommended resources can be web pages, products, topics, or navigation paths that a user might follow, and are presented as hyperlinks in either a separate frame of web pages or a pop-up window. The goal of web recommender systems [LAR02, MGPL04, ZHC04] is to determine which web resources are more likely to be accessed next or in the near future by the current user. In this research, we regard web recommendation as a special kind of web personalization.

1.4 Semantic Web Mining

The current World Wide Web presents a large amount of information in a format tailored for viewing and understanding by humans. People can surf from link to link, query information with search engines, or access websites using domain names. As the retrieved web pages are only meaningful to humans, to the software that processes the contents, they are no more than the strings of random characters. Software programs cannot just load a random document, web page, or file and understand the semantics of that document. Although software could make assumptions based on HTML (HyperText Markup Language) [BL92] or XML (Extensible Markup Language) [YBP⁺04] tags in order to understand web documents, a programmer would have to get involved and determine the meaning, or semantics, of each

tag. From a computer's perspective, the World Wide Web is a confusable, unstructured and less machine-readable mess. As such, a method for representing knowledge, such that software programs can understand, share and exchange knowledge, is needed.

To solve the problem, Berners-Lee *et al.* [BLHL01] proposed the Semantic Web, which is defined as follows: "The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

With the Semantic Web, the large amount of information on the Web can be shared, reused and managed effectively. Since machine can understand the content on the Semantic Web, it enables more advanced automated processing on the Web. Intelligent search engines can be developed to help people find relevant information by using semantic query languages. And new knowledge can be derived from existing information efficiently. As such, many advanced applications and services such as e-business, e-government, and e-learning become possible.

Semantic web mining [BHS02] aims at combining the fast-developing research areas on Semantic Web and web mining. The idea is to improve the results of web mining by exploiting the new semantic structures in the Web, and on the other hand, to make use of web mining techniques for building up the Semantic Web. In this research, we focus on automatic web usage ontology generation for semantic web personalization.

1.5 Objectives

The primary objective of this research is to investigate different web usage mining techniques for discovering the knowledge hidden in web usage logs. In particular, this research focuses on mining web server logs. The discovered knowledge is then applied to some practical web applications such as web recommendation and web personalization. To achieve the objective, we investigate the following issues:

- *Develop new techniques for discovering knowledge from web usage logs.* In particular, we investigate data mining techniques for mining sequential access patterns and association access patterns from web usage logs. The purpose is to develop efficient and effective mining algorithms to discover interesting access patterns from web usage logs. Apart from the common access patterns, we are also interested in mining some special access patterns, such as the periodic access patterns.

Figure 1.3: The proposed research framework.

- *Investigate techniques for applying the discovered knowledge for practical web applications.* The web applications can also serve the purpose to evaluate the quality and effectiveness of the discovered access patterns. In particular, we focus on applications such as web recommendation and web personalization.
- *Investigate new techniques for semantic web usage mining.* We investigate some potential new techniques for extracting semantics from web logs and mining web usage data on the Semantic Web. In particular, we focus on techniques for automatic web usage ontology generation and personalized services on the Semantic Web.

The proposed research framework is shown in Figure 1.3.

1.6 Major Contributions

As a result of this research, we have developed different novel techniques in the areas of pattern discovery and practical web applications listed in Section 1.5. The major contributions of this research are summarized as follows:

- *An Efficient Sequential Pattern Mining Algorithm.* We have proposed CSB-mine (Conditional Sequence Base mining algorithm), an efficient sequential pattern mining algorithm, which aims to discover all sequential patterns with a user-specified minimum support from sequence databases. The CSB-mine algorithm is designed generically so that it can be used by other applications. In this research, we apply it to discover interesting and frequent sequential access patterns from web usage logs.

- *Sequential Web Access based Recommender System (SWARS)*. We have developed a web recommender system, named *SWARS*, to provide personalized web services for assisting users to access related web pages more efficiently and effectively. In the proposed system, a compact tree structure, called Pattern-tree, is used for storing sequential access patterns, matching users' current access sequences and generating web links for recommendations.
- *A Periodic Association Access Pattern Mining Approach*. We have proposed a web usage mining approach for discovering a specific kind of periodic association access patterns of individual users. The proposed approach first uses the fuzzy set theory [Zad65] to represent both real-life temporal concepts and meaningful requested resources, and then, incorporates them into Formal Concept Analysis (FCA) [GW97] for constructing a novel user behavior model, called Personal Web Usage Lattice, from which all periodic association access patterns of the user can be extracted and visualized.
- *Periodic Web Personalization Approach*. We have proposed a Personal Web Usage Lattice based approach for effective periodic web personalization. Different from non-periodic approaches, the proposed approach can determine efficiently which resources a user is most probably interested in during a given time period based on the Personal Web Usage Lattice of the user without using the user's current access information.
- *Semantic Web Personalization Approach*. We have developed an approach for periodic web personalization on the Semantic Web by automatically generating the Web Usage Ontologies from the Web Usage Lattice models and incorporating the periodic web personalization approach into the Semantic Web environment.

Note that all web usage mining algorithms presented in this research do not handle dynamically generated web pages (hidden Web [FLM98] or deep Web [Ber01]) directly. The hidden or deep Web can be handled if other techniques are applied to analyze web content accessed by users and annotate each web access entry in web logs with semantics. One possible approach is to use web content mining techniques to preprocess web logs before using the proposed web usage mining algorithms. However, web content mining is not the main focus in this research.

1.7 Organization of the Thesis

This chapter has introduced the background knowledge and motivations of this research. The objectives of the research have been discussed. We have also listed the contributions that

have been achieved. The rest of the thesis is organized as follows.

Chapter 2 reviews the related work on web usage mining, which includes web usage data, data preprocessing methods and pattern discovery techniques.

Chapter 3 presents the proposed approach, called CSB-mine (Conditional Sequence Base mining algorithm), for mining sequential access patterns. The performance evaluation of the CSB-mine algorithm is also given.

Chapter 4 presents the *SWARS* (Sequential Web Access based Recommender System) for effective web recommendations. The performance evaluation of the proposed system is also given.

Chapter 5 discusses the proposed technique for discovering periodic association access patterns of individual users from web usage logs. The discovered knowledge can be visualized for further understanding the periodic web access behaviors of users.

Chapter 6 discusses the proposed periodic web personalization approach. The Personal Web Usage Lattice proposed in Chapter 5 is used for efficiently generating personalization rules based on given period conditions. The performance of the proposed approach is evaluated based on the quality of the generated personalization rules, and compared with another non-periodic approach.

Chapter 7 presents the approach for developing periodic-based web personalized services on the Semantic Web environment. In this chapter, the automatic Web Usage Ontology generation process is given. We also discuss the Semantic Web services that enable periodic web personalization on the Semantic Web.

Finally, Chapter 8 concludes the thesis with a summary and states the future directions for further research work.

Chapter 2

Web Usage Mining

As mentioned in Chapter 1, web usage mining [SCDT00] aims to discover interesting and frequent user access patterns from web browsing data. The discovered knowledge can then be used for many practical web applications such as web recommendations, adaptive web sites, and personalized web search and surfing. In this chapter, we review the basic concepts and techniques on web usage mining. Firstly, the various types of web usage data are introduced. Next, the major preprocessing tasks are discussed. Then, we review the different pattern discovery techniques for web usage mining. Finally, a summary of the chapter is given.

2.1 Web Usage Data

Web usage data record access patterns of users from websites. However, it can also include data from user profiles, registration information, cookies, user queries, bookmarks and any other data derived from the interactions of users while surfing on the Web. Web usage data are mainly divided into three types, namely web server logs, proxy server logs and client browser logs.

The web server logs contain the most important data for web usage mining. This type of logs records the access of a website by multiple users. Each user access record contains the client IP address, request time, requested URL, HTTP status code, etc. These log files can be stored in various standard formats, such as the common log file format, extended log file format, etc. A fragment of a web server log from “North Latitude One BBS” (<http://bbs.nlone.net>) in the W3C extended log file format is given in Figure 2.1.

```

155.69.220.183 - - [01/May/2005:10:58:54 +0800] "GET /bbscon.php?board=MovieTV&id=11810 HTTP/1.1" 200 8388
155.69.220.183 - - [01/May/2005:10:58:54 +0800] "GET /bbscon.js HTTP/1.1" 304 -
155.69.220.183 - - [01/May/2005:10:58:54 +0800] "GET /images/reply.gif HTTP/1.1" 304 -
155.69.59.119 - - [01/May/2005:10:59:35 +0800] "GET /bbscon.php?board=photo&id=27341 HTTP/1.1" 200 4083
155.69.59.119 - - [01/May/2005:10:59:37 +0800] "GET /cgi-bin/bbs/bbstcon?board=photo&gid=27330 HTTP/1.1" 200 3525
155.69.59.119 - - [01/May/2005:10:59:38 +0800] "GET /jscon.php?bid=62&id=27330 HTTP/1.1" 200 1888
155.69.59.119 - - [01/May/2005:10:59:41 +0800] "GET /ansi-web-middle.css HTTP/1.1" 304 -
155.69.59.119 - - [01/May/2005:10:59:42 +0800] "GET /images/files/img.gif HTTP/1.1" 200 275

```

Figure 2.1: A fragment of a web server log from “North Latitude One BBS”.

A web proxy server serves as a gateway between client browsers and web servers. Proxy caching can be used for reducing the loading time of a web page accessed frequently by users as well as the network traffic loading at the server and client sides. Proxy server can trace all HTTP requests from multiple clients to multiple web servers. From proxy server logs, we can also analyze the browsing behavior of a group of anonymous or identifiable users sharing a common proxy server.

Client-side usage data collection can be implemented by using an agent installed at a client or web browser with the capability of tracing web access activities of users. Client-side logs can collect usage data about a single user over multiple websites. Client-side data collection can capture more information than the web server or proxy server logs, such as mouse clicks on the refresh or back button of the web browser.

In this section, we focus our discussion on web server logs, which are the most commonly used data for web usage mining. Nevertheless, most of the web mining techniques discussed here can also be applied to other types of web usage data in a similar manner.

2.2 Preprocessing

Generally, web usage logs can be regarded as a collection of sequences of access events from one user or session in timestamp ascending order. However, the original web usage logs probably are not in a format that is usable for mining purposes. So such data may need to be cleaned and reformatted. The preprocessing step aims to preprocess the original web usage logs to identify all user access sessions or convert the web usage logs into other formats. The main preprocessing tasks [CMS99] include data cleaning, user identification and session identification and other advanced preprocessing.

2.2.1 Data Cleaning

In the original web usage logs, not all records are valid for web usage mining. We only treat requested resources as useful when they are in HTML format. Therefore, apart from records

such as URLs of HTML or extended HTML documents (e.g., xhtml, ASP, PHP or JSP), all other records are discarded from the web usage logs. These include records containing URLs of image files, such as GIF, JPG or BMP files. HTTP status codes are used to indicate the success or failure of the requested event. Only records with codes between 200 and 299 are considered as successful records, and others are discarded from the web usage logs. Moreover, requests from automatic agent software, robots and Web spiders or crawlers need to be identified and removed as well [TK02], since such requests are usually significantly different from the surfing characteristics of humans.

2.2.2 User Identification

For analyzing users' access behaviors, unique users must be identified. For web server logs, all users' access activities of a website are recorded by the Web server of the website. Users are treated as anonymous since the IP addresses are not mapped to any user-identifiable profile databases. In this case, we can simplify user identification to client IP identification. In other words, requests from the same IP address can be regarded as from the same user and stored into the same group under that user. In order to identify users more accurately, some other information from web usage logs may be helpful. For example, the *agent* recorded in web usage logs contains information on the client browser and operating system. Since different users may access the website through the same proxy server, the IP address may be the same. However, the agent type may not be the same in many cases. Therefore, it is quite reasonable to assume that each different agent type for the same IP address represents a different user.

2.2.3 Session Identification

For logs from a user that spans a long period of time, it is very likely that the user has visited the website more than once. The goal of session identification is to divide web logs of each user into individual access sessions. The simplest method is to set a timeout threshold. If the difference between the request time of two adjacent records from a user is greater than the timeout threshold, it can be considered that a new access session has started. In this research, we use 30 minutes as the default timeout threshold. Recently, more accurate session identification approaches have been proposed. For example, Huang *et al.* [HPAS04] presented a novel session identification method based on statistical language models. It uses an information theory approach to detect session boundaries dynamically to achieve more

accurate results for identifying sessions. Lou *et al.* [LLLY02] proposed a more advanced cut-and-pick method for determining the access sessions from proxy logs by deciding on more reasonable session boundaries and removing noisy accesses.

2.2.4 Other Preprocessing Tasks

For some usage mining purposes, further preprocessing tasks [CMS99] may be applied. *Path completion* is used to find the actual access paths among web pages. The *referrer* field in the web usage logs can be checked to find out which page the request has come from. If the referrer is unavailable, the link structure of the website can also help to estimate the access paths of users. The goal of *transaction identification* is to create meaningful clusters of requested web pages for each user. Therefore, the task of transaction identification is to divide a larger transaction into multiple smaller ones or merge smaller transactions into larger ones. Some transaction identification approaches such as the *reference length* [CMS99] and *maximal forward reference* [CPY96] have been proposed for preprocessing web usage logs.

2.3 Pattern Discovery Techniques

Various data mining techniques [SCDT00] have been investigated for mining web usage logs. They are statistical analysis, association rule mining, clustering, classification and sequential pattern mining.

2.3.1 Statistical Approaches

Statistical techniques are the most commonly used methods for extracting knowledge from web usage logs. The useful statistical information discovered from web usage logs is listed in Table 2.1.

The types of statistical information shown in Table 2.1 are usually generated periodically in reports and used by administrators for improving the system performance, facilitating the website modification task, enhancing the security of the system, and providing the support for marketing decisions. Many web traffic analysis tools, such as WebTrends [Web] and SurfAid [Sur], are available for generating web usage statistics.

2.3.2 Association Rule Mining

Association rule mining finds interesting association or correlation relationships among a large set of data items. A typical example of association rule mining is market basket analysis.

Table 2.1: Useful statistical information discovered from web logs.

Statistics	Detailed Information
Website activity statistics	Total number of visits Average number of hits Successful/failed/redirected/cached hits Average view time Average length of a path through a site
Diagnostic statistics	Server errors Page not found errors
Server statistics	Top pages visited Top entry/exit pages Top single access pages
Referrers statistics	Top referring sites Top search engines Top search keywords
User demographics statistics	Top geographical location Most active countries/cities/organizations
Client statistics	Visitor's web browser, operating system, and cookies

This process analyzes customer buying habits by finding associations between the different items that customers place in their “shopping baskets”. The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers. Apriori [AS94] is a classical algorithm for mining association rules. Some variations of the Apriori approach for improving the efficiency of the mining process are referred to as Apriori-based mining algorithms. FP-growth [HPY00] is an efficient approach for mining frequent patterns without candidate generation.

For web usage mining, association rules can be used to find correlations between web pages (or products in an e-commerce website) accessed together during a session. Such rules indicate the possible relationships between pages that are often viewed together even if they are not directly connected, and can reveal associations between groups of users with specific interests. Apart from being exploited for business applications, the association rules can also be used for web recommender systems, web personalization, or improving the system's performance through predicting and pre-fetching of web data.

In [LAR02], Lin *et al.* proposed a collaborative recommendation technique based on an adaptive-support algorithm for mining association rules. The algorithm does not require the minimum support to be specified in advance. Given a range for the number of rules, it can automatically adjust the minimum support in order to obtain a subset of rules whose size lies within the desired range. This strategy can reduce the runtime of the mining process and

provide enough rules for achieving good recommendation performance. However, the target range is more difficult to be specified reasonably than the minimum support if the user has no prior knowledge on user transactions. For recommendations, two kinds of rules, namely user associations and article associations, are mined and combined together. The performance of this approach is significantly better than that of traditional correlation-based methods.

In [MGPL04], Moreno *et al.* proposed a web usage mining approach for making recommendations based on a predictive model built from association rules between user and product attributes. A refinement method based on the concept of unexpectedness is used to obtain stronger rules that reinforce the relations between items and minimize the recommendation errors. The recommendation procedure aims to search for the best matching rules from the set of refined association rules in order to recommend products.

In [MDLN01], Mobasher *et al.* proposed a scalable framework for web personalization based on association rules mined from web usage data. Firstly, all frequent itemsets are mined from user access transactions using the Apriori algorithm [AS94]. The discovered itemsets are stored in a directed acyclic graph, called Frequent Itemset Graph, which is an extension of the lexicographic tree used in the tree projection algorithm [AAP01]. Given a current active user session, stored in lexicographic order under a sliding window of size w , a depth-first search of the Frequent Itemset Graph is performed from level w . If a match is found, the children of the matching node generate candidate recommendations. All candidate rules satisfying the given minimum confidence threshold form the recommendation set with their confidence values as the recommendation scores. The use of Frequent Itemset Graph makes it efficient to produce recommendations in real-time without the need to generate all association rules from frequent itemsets. To improve the performance of recommendations, multiple support levels for different types of pages and user histories are also used. The proposed approach has achieved better performance than the kNN (k-Nearest-Neighbors) based collaborative filtering approaches both in terms of coverage and accuracy of recommendations.

In [WSP01], Wong *et al.* proposed a new approach to improve the quality of web access pattern prediction by combining fuzzy association rule mining and CBR (case-based reasoning) [SP04]. One of the problems of traditional association rule mining techniques is that each item is only considered to either exist in a transaction or not. Therefore, the user's preference and interest on each item of a transaction cannot be presented. Since the concept of preference and interest are fuzzy data, the fuzzy logic techniques [Zad75] are applied. The time duration of each page view is selected as the fuzzy set attribute of a web access case

and fuzzified into five linguistic terms: Long, Quite Long, Medium, Quite Short, and Short. The number of fuzzy association rules is usually larger than traditional association rules. To improve the efficiency of rule matching, the fuzzy index tree technique is developed to group fuzzy rules together by merging the fuzzy values. The fuzzy rule set has performed better in prediction accuracy and rule coverage compared with the traditional rule set.

Yang *et al.* [YP02] proposed a scalable web usage mining approach for predicting web accesses based on association rules that are temporally constrained and ordered. The main idea relies on the simple premise that more recently accessed web pages are likely to have a greater influence on web pages that will be accessed in the near future. Firstly, the extracted association rules are ordered based on their sequential dominance and confidence. Then, a rule based predictive model uses the top few rules to predict the set of web pages that will be accessed by users in the near future. The proposed approach has not only achieved good results for predictions, but also good performance in terms of space and time complexity by pruning the rule-space significantly. This makes it possible for more efficient online prediction.

2.3.3 Sequential Pattern Mining

Sequential pattern mining, which discovers frequent patterns from sequence databases, is defined as follows [AS95]. Given a sequence database where each sequence is a list of transactions ordered by transaction time and each transaction consists of a set of items, find all sequential patterns with a user-specified minimum support, where the support is the number of data sequences that contains the pattern. Sequential pattern mining techniques are mainly based on three approaches: Apriori-based mining algorithms, projection-based mining algorithms and WAP-tree (Web Access Pattern tree) based mining algorithms.

Apriori-based Mining Algorithms

Most of the basic and earlier algorithms for sequential pattern mining are based on the Apriori property proposed in association rule mining [AS94]. Such property states that any sub-patterns of a frequent pattern must be frequent. Based on this heuristic, various Apriori-based sequential pattern mining algorithms have been proposed. The AprioriAll [AS95] algorithm proposed a three-step approach for mining sequential patterns. It first finds all frequent itemsets. Then, it transforms the database such that each original transaction is replaced by the set of all frequent itemsets contained in the transaction. And finally, it finds the sequential patterns. However, this algorithm does not scale well due to the costly

transformation step. In [SA96], a generalized sequential pattern mining algorithm known as GSP (Generalized Sequential Pattern) mining algorithm was proposed. Similar to the AprioriAll algorithm, GSP scans the database several times. In the first scan, it finds all frequent items and forms a set of frequent sequences of length one. In subsequent scans, it generates candidate sequences from a set of frequent sequences obtained from the previous scan and checks their supports. The process terminates when no candidate is found to be frequent.

Projection-based Mining Algorithms

The Apriori-based sequential pattern mining approaches may substantially reduce search space. However, such approaches encounter the efficiency problem when a sequence database is large and/or when sequential patterns to be mined are numerous and/or long. A number of data projection-based algorithms have been proposed for mining sequential patterns more efficiently. The main idea is that sequence databases are recursively projected into a set of smaller projected databases and sequential patterns are grown in each projected database by exploring only local frequent fragments. In particular, two pattern growth schemes, FreeSpan (**F**requent pattern-projected **S**equential **p**attern mining) [HPMA⁺00] and PrefixSpan (**P**refix-projected **S**equential **p**attern mining) [PHMA⁺01], were proposed. They mine the complete set of sequential patterns but greatly reduce the efforts of candidate subsequence generation. To further improve mining efficiency, three kinds of database projections, namely level-by-level projection, bi-level projection and pseudo-projection, are explored. FreeSpan and PrefixSpan have outperformed the Apriori-based GSP algorithm and an integrated PrefixSpan has achieved the best performance in mining large sequence databases.

WAP-tree based Mining Algorithms

Apart from the projection-based algorithms, another series of algorithms that avoid generating candidate sequences are known as WAP-tree based mining algorithms. The WAP-tree [PHMAZ00] is a very effective compressed data structure designed for storing the data obtained from web logs. The WAP-tree is constructed over the frequent sub-sequences in the web access sequence database by merging their common prefixes. At the same time, all nodes that contain the same frequent event are linked into an event queue and the Header Table with all frequent events is created for this WAP-tree with the head of each event queue registered in it. Then, all nodes labeled with the same event can be visited by following the

related event queue, starting from the Header Table.

The WAP-mine algorithm [PHMAZ00] was proposed to mine sequential patterns from the WAP-tree. The approach uses a heuristic, called conditional search, to narrow the search space efficiently by looking for patterns with the same suffix and thus avoids generating large candidate sets as in Apriori-based algorithms. The WAP-mine algorithm processes each event one by one. For event e_i , it firstly forms the conditional WAP-tree for it, which contains the set of prefixes of the subsequences that contain e_i as a suffix. After that, the algorithm continues to mine the conditional WAP-tree recursively. Finally, the results obtained from mining the conditional WAP-tree are concatenated with e_i . Sequences obtained through this algorithm correspond to the final sequential access patterns.

Currently, there are a few extensions of the WAP-tree and the corresponding algorithms for mining sequential access patterns. In [EL05], Ezeife and Lu proposed the Pre-Order Linked WAP-Tree Mining (PLWAP) algorithm with Tree Binary Code Assignment rules which find common prefix patterns instead of suffix patterns as used in the WAP-mine algorithm. This avoids recursively re-constructing intermediate conditional WAP-trees by assigning the binary position codes to each node in the WAP-tree. The FS-tree [MCE04] extended the WAP-tree structure for incremental and interactive mining. The corresponding mining algorithm FS-mine (Frequent Sequence mining) was proposed to analyze the FS-tree to discover frequent sequences. In [ZHF04], an efficient WAP-tree based mining algorithm, called CS-mine (Conditional Sequence mining algorithm) was proposed. It has achieved better performance than the WAP-mine algorithm by eliminating the need for the re-construction of intermediate conditional WAP-trees. However, it also needs to construct an initial WAP-tree from the web access sequence database. Apart from extending the WAP-tree and the corresponding mining algorithms, the WAP-tree structure has also been used by some web applications [PSR03].

For web usage mining, a sequential web access pattern [PHMAZ00] is a sequential pattern in a large set of pieces of web usage logs, which is pursued frequently by users. Such knowledge can be used for discovering useful user interest trends and predicting future accesses, which is helpful for pre-fetching documents, recommending web pages, or placing advertisements aimed at certain user groups.

In [MDLN02], Mobasher *et al.* studied the problem of web personalization based on sequential and non-sequential patterns discovered from web usage logs. It has shown that more restrictive patterns, such as contiguous sequential patterns (frequent navigational paths), are

more suitable for predictive tasks such as web pre-fetching, in which the primary goal is to predict which item will be accessed next by a user. On the other hand, less constrained patterns, such as frequent itemsets or general sequential patterns are more effective alternatives in the context of web personalization and recommendation systems.

Gery and Haddad [GH03] also presented a framework for a recommender system in order to evaluate three web usage mining approaches that use association rules, sequential patterns and generalized sequential patterns (sequential patterns with wildcards) for predicting a user's next request. It has shown that the recommendation approach with sequential patterns has achieved better accuracy than that with association rules and generalized sequential patterns when one-step prediction is used.

2.3.4 Clustering

Clustering [JMF99] is a technique for grouping a set of physical or abstract objects into classes of similar objects. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. A cluster of data objects can be treated collectively as one group in practical applications. The choice of a clustering algorithm depends both on the type of data available, and on its purpose and application.

For web usage mining, clustering techniques [XP01, MCS99, NFJK99, BG00, WAS05, KJ05] are mainly used to discover two kinds of useful clusters, namely user clusters and page clusters. User clustering attempts to find groups of users with similar browsing preference and habit, whereas web page clustering aims to discover groups of pages that seem to be conceptually related according to the users' perception. Such knowledge is useful for performing market segmentation in e-commerce and web personalization applications.

In [XP01], Xie *et al.* presented a novel web usage mining approach for clustering Web users into different groups and generating common user profiles. The proposed clustering technique uses the belief function based on Dempster-Shafer's theory to capture the uncertainty among user access behavior. The generated profiles can be used for web recommendations, personalized websites, and targeting users for personalized advertising.

In contrast to clustering user sessions, Mobasher *et al.* [MCS99] introduced an effective clustering technique by directly computing overlapping clusters of URL references based on their co-occurrence patterns across user transactions. This method can obtain clusters that potentially capture overlapping interests of different types of users, even if these transactions

are not deemed to be similar. As traditional clustering techniques such as distance-based methods are generally unable to handle this type of clustering, the Association Rule Hypergraph Partitioning (ARHP) [HKKM98] technique was applied for clustering. Since ARHP provides automatic filtering capabilities, it does not require distance computations, and can be used in high-dimensional datasets without requiring dimension reduction. Once the URL clusters have been computed, the current active session can be assigned to the best matching cluster by computing the matching score of each cluster. Then, the recommendation score is computed for each URL in the matching cluster, and all URLs whose scores satisfying a minimum recommendation threshold are returned as recommended links.

The web access sessions are too complex to be converted into simple numerical features. In fact, the URLs in a website always have a hierarchical or tree-like structured directory. In [NFJK99], Nasraoui *et al.* defined a new similarity measure between two web access sessions that incorporates both the hierarchical organization of the website and the URLs involved. This session similarity measure is not Euclidean. The fuzzy Competitive Agglomeration for Relational Data (CARD) clustering algorithm [FK97] was proposed to automatically cluster data into parsimonious number of components to analyze web server logs and obtain a typical session profile of users.

In [BG00], Banerjee and Ghosh discussed two session extraction techniques for web logs, namely time-out heuristic approach and link following heuristic approach, and based on the extracted sessions, it proposed a novel and effective algorithm for clustering web users. The purpose of clustering users based on the access paths is to find groups of users with similar interests and motivations for visiting the website. A Min-Max Path Similarity measure between any pairs of sessions is defined for mapping the sessions into a similarity space. For large websites, if the access paths are considered at a web page level, many similarity values are zero since very few access paths have actual page overlaps. To handle this problem, the web pages are first grouped into concept categories based on suitable metadata information. After converting the raw access paths into concept-category based access paths, the average size of paths is reduced and the meaning of paths can be easily understood. Once these paths are formed, an efficient hypergraph partitioning algorithm can be used for clustering. The clustering results are more meaningful than the simple page-based path clustering. However, since the categorization of web pages may be very complicated for websites that are not well organized, the applicability of this approach is limited.

In [WAS05], Wang *et al.* proposed a hybrid neuro-fuzzy approach for web traffic mining

and prediction. The proposed approach uses a clustering algorithm, Self Organizing Map (SOM) [Koh88], to discover hidden access patterns from Web server logs. The clustered data and clustering information are further used for learning the trend patterns by using statistical analysis methods. In order to make the analysis more intelligent the clustered data are used to predict the long-term (daily) and short-term (hourly) traffic including request volume and page volume using the fuzzy inference method.

Kamdar *et al.* [KJ05] developed an adaptive Web server that generates pages online based on a user's past traversal patterns mined using a fuzzy incremental clustering algorithm. Over time, a user's traversal pattern changes and the information in the user profile is updated incrementally to reflect the changes to the personalized page generated for the user.

2.3.5 Classification

Classification is the process of building a model to classify a class of objects so as to predict the class label of a future object whose class is not known. Since the class label of each training sample is provided, this process is also known as *supervised learning* (i.e., the learning of the model is "supervised" in that it is told to which class each training sample belongs).

For web usage mining, classification is usually used to construct profiles of users belonging to a particular class or category. There is not much work done using classification methods directly for web usage mining due to the complexity of web usage data.

Tan *et al.* [TK02] examined the problem of identifying web robot sessions using standard classification techniques. The goal is to determine the minimum number of requests needed to distinguish between robot and non-robot sessions from web server logs with high accuracy. After preprocessing the web server logs to identify access sessions, a set of features is extracted to characterize the access session. These features include total pages, total time, average time, request method, request error and so on, which can be derived easily from the web server logs. Then, all training sessions are labeled as robot or non-robot sessions using the class label. The C4.5 algorithm [Qui93] is used to construct the classification model.

In [CKK02], Cho *et al.* proposed a personalized recommender system based on web usage mining techniques. Classification is used to minimize recommendation errors by making recommendation only for customers who are likely to buy the recommended products. The proposed approach first learns the customer preference and product association automatically from web usage data. Next, customers who are likely to buy the recommended products are selected using decision tree induction. Then, expert knowledge such as product taxonomy

is incorporated into the recommendation process. Finally, highly business-efficient products among the candidate recommendable products are chosen using the proposed measures.

2.4 Summary

In this chapter, we have reviewed the related work on web usage mining including web usage data, preprocessing tasks, and various pattern discovery techniques.

Web usage data is the main source for web usage mining, which mainly includes web server logs, proxy server logs and client browser logs. Since web server logs have standard formats and are available in all web servers, they are most widely used in research on web usage mining. For web usage data preprocessing, the tasks include data cleaning, user identification and session identification. The fundamental techniques for discovering patterns in web usage logs include statistical analysis, association rule mining, clustering, classification and sequential pattern mining. Statistical techniques are generally utilized for extracting statistical knowledge from web logs. Such knowledge is most useful for analyzing web traffic of a website. Association rule mining can be used for finding related pages that are most often referred together in an access session. Sequential pattern mining is an extension of association rule mining by considering the order of occurrence of items in transactions. Sequential patterns are sequences of web pages accessed frequently by users. Such patterns are useful for discovering user behaviors and predicting future pages to be visited by the user. Clustering techniques can be used to discover page clusters and user clusters from web logs. Page clusters are useful for improving search engines and providing web categorization, whereas user clusters are useful for inferring user demographics in order to provide personalized web contents to users. Classification is the task of predicting a data item into one of the several predefined classes. These classes usually represent different user profiles, and classification is performed based on the features that best describe the properties of a given class or category. The main pattern discovery techniques are summarized in Table 2.2.

Table 2.2: Pattern discovery techniques for web usage mining.

Author	Technique	Approach	Application
WebTrends [Web]	Statistics	Statistical approach	Business intelligence
SurfAid [Sur]	Statistics	Statistical approach	Business intelligence
Lin <i>et al.</i> [LAR02]	Association rule mining	Adaptive-support	Collaborative recommender systems, E-commerce
Moreno <i>et al.</i> [MGPL04]	Association rule mining	Incremental mining	Personalized recommendations
Mobasher <i>et al.</i> [MDLN01]	Association rule mining	Frequent Itemset Graph	Web personalization
Wong <i>et al.</i> [WSP01]	Association rule mining	Fuzzy theory, Case-based reasoning, Fuzzy index tree	Web access path prediction, Web personalization
Yang <i>et al.</i> [YP02]	Association rule mining	Temporal constraint	E-commerce, Recommendation systems
Pei <i>et al.</i> [PHMAZ00]	Sequential pattern mining	WAP-tree, WAP-mine	General
Ezeife <i>et al.</i> [EL05]	Sequential pattern mining	Tree binary code assignment, PLWAP	General
Maged <i>et al.</i> [MCE04]	Sequential pattern mining	Incremental mining, FS-tree, FS-Miner	Web pages predicting and Pre-fetching
Prakash <i>et al.</i> [PSR03]	Sequential pattern mining	WAP-tree	Personalized surfing system
Mobasher <i>et al.</i> [MDLN02]	Association rule mining, sequential pattern mining	Frequent itemsets, general sequential patterns, contiguous sequential patterns	Web personalization and recommender systems, Web prefetching
Gery <i>et al.</i> [GH03]	Association rule mining, sequential pattern mining	Association rules, sequential patterns, generalized sequential patterns	Recommender systems
Xie <i>et al.</i> [XP01]	Clustering	Web user clustering	Web personalization
Mobashe <i>et al.</i> [MCS99]	Association rule mining, clustering	Association Rule Hypergraph Partitioning	Web personalization
Nasraoui <i>et al.</i> [NFJK99]	Clustering	Fuzzy competitive agglomeration clustering	Web personalization
Banerjee <i>et al.</i> [BG00]	Clustering	Concept-based clustering	Web personalization
Wang <i>et al.</i> [WAS05]	Clustering	Hybrid Neuro-Fuzzy approach, Self Organizing Map	Web traffic trend predictions
Kamdar <i>et al.</i> [KJ05]	Clustering	Incremental fuzzy clustering	E-commerce, Web personalization
Tan <i>et al.</i> [TK02]	Classification	Decision tree	Preprocessing
Cho <i>et al.</i> [CKK02]	Classification	Decision tree	Personalized recommender systems

Chapter 3

Efficient Sequential Access Pattern Mining

A sequential access pattern [PHMAZ00] is a sequential pattern in a large set of pieces of web logs, which is pursued frequently by users. Most of the previous studies for discovering sequential patterns such as AprioriAll [AS95] and GSP [SA96] are mainly based on the Apriori algorithm [AS94]. However, these algorithms encounter the same problem as most Apriori-based algorithms that require expensive multiple scans of databases in order to determine which of the candidates are actually frequent.

To improve the efficiency of sequential pattern mining algorithms, Pei *et al.* [PHMAZ00] proposed a novel and highly compressed data structure known as Web Access Pattern Tree (or WAP-tree) which is based on the FP-tree [HPY00] structure with the consideration on the sequential characteristics of items. The WAP-tree structure facilitates the development of novel algorithms for mining sequential access patterns efficiently from a large set of web log pieces. In particular, the WAP-mine algorithm [PHMAZ00] was developed for mining sequential access patterns from the WAP-tree. This approach avoids the problem of generating an explosive number of candidates as encountered in Apriori-based algorithms. And experimental results have shown that the WAP-mine algorithm is in general an order of magnitude faster than traditional sequential pattern mining techniques. This can be attributed to the compact structure of the WAP-tree and the novel conditional search strategies used in the WAP-mine algorithm. However, the use of conditional search strategies in the WAP-mine algorithm requires the re-construction of large numbers of intermediate conditional WAP-trees during the mining process, which is also very costly.

In this chapter, we propose an efficient sequential access pattern mining algorithm called

CSB-mine (Conditional Sequence Base mining algorithm). The proposed CSB-mine algorithm is based directly on the conditional sequence bases and eliminates the need for the construction of the initial WAP-tree, and the re-construction of costly intermediate conditional WAP-trees as in the WAP-mine algorithm. The main idea is described as follows. A conditional sequence base is a set of all suffix sequences of sequences with the same specific sequential pattern as their prefix sequence. In the WAP-mine algorithm, the conditional sequence bases of 1-length sequential patterns (i.e., frequent events) are generated from the initial WAP-tree. And then, the conditional WAP-trees of n -length ($n \geq 1$) sequential patterns are constructed from the conditional sequence bases of n -length sequential patterns. Subsequently, the conditional WAP-trees of n -length sequential patterns are used for generating the conditional sequence bases of $(n+1)$ -length sequential patterns. The recursive constructions of sub-conditional sequence bases and sub-conditional WAP-trees will be stopped until the new conditional WAP-trees has only one branches or the new conditional sequence bases are empty. Actually, the conditional sequence bases of $(n+1)$ -length sequential patterns can be computed from the conditional sequence base of n -length sequential patterns directly without the intermediate conditional WAP-tree of n -length sequential patterns. Therefore, the efficiency of the WAP-mine algorithm can be further improved by avoiding the costly constructions of the initial and all conditional WAP-trees. In addition, the proposed CSB-mine algorithm has the similar idea to a projection-based approach, PrefixSpan [PHMA⁺01], which generates sequential access patterns by recursively projecting the suffix sequences of local frequent prefix sequences into small projected databases. In subsequent sections, we will also show that the proposed CSB-mine algorithm outperforms the PrefixSpan algorithm in terms of efficiency.

In Chapter 4, we will discuss a sequential access pattern based web recommender system to illustrate the use of the CSB-mine algorithm for a practical web application.

The rest of this chapter is organized as follows. In Sections 3.1 and 3.2, we introduce some basic concepts on sequential access pattern mining and the WAP-mine algorithm. The proposed CSB-mine algorithm is then presented in Section 3.3. The performance of the CSB-mine algorithm is evaluated in Section 3.4. Finally, a summary of this chapter is given in Section 3.5.

Table 3.1: A sample database of web access sequences.

User ID	Web Access Sequence
1	<i>abdac</i>
2	<i>eaebcac</i>
3	<i>babfae</i>
4	<i>afbaccfc</i>

3.1 Problem Statement

Generally, web logs can be regarded as a collection of sequences of access events from one user or session in timestamp ascending order. Preprocessing tasks [CMS99] including data cleaning, user identification and session identification can be applied to the original web log files to obtain the web access sequences.

Let E be a set of unique access events, which represents web resources accessed by users, i.e., web pages, URLs or topics. For example, for a static website, the set of access events consists of the URLs of all static web pages, but for a dynamic website, the set of access events consists of the dynamic URLs of web pages (i.e., URLs with input parameters). If each URL can be categorized into one predefined topic, such as News, Sports and Entertainment, we can also use the set of predefined topics as the set of access events. A web access sequence $S = e_1e_2\dots e_n$ ($e_i \in E$ for $1 \leq i \leq n$) is a sequence of access events, and $|S| = n$ is called the *length* of S . Note that it is not necessary that $e_i \neq e_j$ for $i \neq j$ in S , that is repeat of items is allowed. All web access sequences in a database can belong to either a single user (for client-side logs) or multiple users (for server and proxy logs). Suppose we have a set of web access sequences with an access event set, $E = \{a, b, c, d, e, f\}$. A sample web access sequence database is given in Table 3.1.

In $S = e_1e_2\dots e_ke_{k+1}\dots e_n$, $S_{prefix} = e_1e_2\dots e_k$ is called a *prefix sequence* of S , or a *prefix sequence* of e_{k+1} in S . And $S_{suffix} = e_{k+1}e_{k+2}\dots e_n$ is called a *suffix sequence* of S or a *suffix sequence* of e_k in S . A web access sequence can be denoted as $S = S_{prefix} + S_{suffix}$. For example, $S = abdac$ can be denoted as $S = a + bdac = ab + dac = \dots = abda + c$. Let S_1 and S_2 be two suffix sequences of e_i in S , and S_1 is also the suffix sequence of e_i in S_2 . Then, S_1 is called the *sub-suffix sequence* of S_2 and S_2 is the *super-suffix sequence* of S_1 . The suffix sequence of e_i in S without any super-suffix sequence is called the *long suffix sequence* of e_i in S . For example, if $S = abdacb$, then $S_1 = cb$ is the sub-suffix sequence of $S_2 = bdacb$ and S_2 is the super-suffix sequence of S_1 . S_2 is also the long suffix sequence of a in S . Given a web access sequence database $WAS_{DB} = \{S_1, S_2, \dots, S_m\}$ in which S_i ($1 \leq i \leq m$) is a web

access sequence, and $|WAS_{DB}| = m$ is called the *size* of the database. The *support* of S in WAS_{DB} is defined in the following equation:

$$Sup(S) = \frac{|\{S_i | S \subseteq S_i, S_i \in WAS_{DB}\}|}{|WAS_{DB}|}$$

For example, let's consider the sample database in Table 3.1, then $Sup(abac) = \frac{3}{4} = 75\%$ and $Sup(baba) = \frac{1}{4} = 25\%$. A web access sequence S is called a *sequential access pattern*, if $Sup(S) \geq MinSup$, where $MinSup$ is a given support threshold. For the sample database, suppose $MinSup = 75\%$, then it is required to find all sequential access patterns supported by at least 75% web access sequences from the sample database, such as abc , $abac$, etc.

3.2 The WAP-tree and the WAP-mine Algorithm

The WAP-tree is a very effective compressed data structure designed for storing web access sequences. To construct a WAP-tree, we need two scans of the web access sequence database: (1) Scan WAS_{DB} once, find all frequent individual events; (2) Scan WAS_{DB} again, construct a WAP-tree over the sub-sequences with only frequent individual events of each web access sequence, which are also called *frequent sub-sequences*, by merging their common prefixes.

For example, the first scan of the database in Table 3.1 derives the set of frequent individual events, i.e., $\{a, b, c\}$. The frequent sub-sequences are $\{abac, abcac, baba, abacc\}$. The WAP-tree of the web access sequence database in Table 3.1 is shown in Figure 3.1, which is constructed as follows. First, insert the sequence $abac$ into the initial tree with only one virtual root node *Root*. It creates a new node ($a : 1$) (i.e., labeled as a , with count set to 1 initially) as the child of the root, and inserts this node into the event queue of a , i.e., a -queue. Then, it derives the a -branch “($a : 1$) $\rightarrow$ ($b : 1$) $\rightarrow$ ($a : 1$) $\rightarrow$ ($c : 1$)”, in which arrows are used to point from parent nodes to children, and inserts each node in the related event queue respectively. Second, insert the sequence $abcac$ starting from the root node. Since the root has a child labeled a , a 's count is increased by 1, i.e., ($a : 2$). Similarly, we have ($b : 2$). The next event, c , does not match the existing node a , and then, a new child node ($c : 1$) is created and inserted into the WAP-tree and c -queue at the same time. The remaining processes can be derived accordingly. Finally, the Header Table with all frequent events is created for this WAP-tree with the head of each event queue registered in it. Then, all the nodes labeled with the same event can be visited by following the related event queue starting from the Header Table. For example, we can access all the nodes labeled a by following the a -queue

Figure 3.1: The WAP-tree for frequent sub-sequences in Table 3.1.

“(a : 3) → (a : 2) → (a : 1) → (a : 1)” starting from the record *a* in the Header Table.

The foundation of the WAP-mine algorithm [PHMAZ00] is based on a heuristic called *Suffix Heuristic*, i.e., if an event e_i is frequent in the prefixes of sequences that have a suffix which contains sequential pattern P as a subsequence, then $e_i + P$ is also a sequential access pattern.

The main steps involved in the WAP-mine algorithm are listed as follows:

1. Construct a conditional sequence base of each event in WAP-tree by following the event-queue, count the conditional frequent events at the same time.
2. Build an intermediate conditional WAP-tree over the conditional sequence base found in the previous step.
3. Recursively repeat steps 1 and 2, until the constructed conditional WAP-tree has only one branch or the set of conditional frequent events is empty.
4. Return all the unique combinations of nodes in all conditional WAP-trees with a single branch as the complete sequential access patterns.

For example, the conditional WAP-tree of *a* is shown in Figure 3.2(a). Since it is not a single branch WAP-tree, recursive construction of the conditional WAP-tree of each frequent event is needed. For the conditional frequent event *b*, we can construct the conditional WAP-tree of *ba* shown in Figure 3.2(b). It is already a single branch WAP-tree, and the sequential access patterns with the suffix *ba*, i.e., $\{aba, ba\}$, can be extracted.

As can be seen in the example, the WAP-mine algorithm narrows the search space efficiently, and avoids the overwhelming problems of generating explosive candidates in Apriori-based algorithms. However, the construction of the initial WAP-tree and the re-construction of intermediate conditional WAP-trees during mining are also very costly.

Figure 3.2: Conditional WAP-trees for WAP-tree in Figure 3.1.

3.3 The CSB-mine Algorithm

The CSB-mine (Conditional Sequence Base mining algorithm) employs directly the conditional sequence base of each frequent event, without the need of constructing any WAP-trees. Although the WAP-tree is a highly compressed data structure for storing sequence data, we need recover the uncompressed sequences during the mining process. This is the main reason why our CSB-mine algorithm is not based on the WAP-tree. As such, the construction of the initial WAP-tree and re-construction of large numbers of intermediate conditional WAP-trees are no longer required. Therefore, the CSB-mine algorithm can improve quite significantly on efficiency when compared with the WAP-mine algorithm.

Figure 3.3 gives an overview of the proposed CSB-mine algorithm which consists of the following steps:

1. *Preprocessing.* Construct the initial conditional sequence base from the web access sequence database.
2. *Constructing Event Queues for Conditional Sequence Base.* Construct event queues for the current conditional sequence base, so that all the same conditional frequent events in the current conditional sequence base can be visited one by one by following the event queues.
3. *Single Sequence Testing for Conditional Sequence Base.* Check whether all sequences in the current conditional sequence base can be combined into a single sequence. If so, such single sequence will be used to form a part of the sequential access patterns, and the mining of the current conditional sequence base will be stopped.
4. *Constructing Sub-Conditional Sequence Base.* Construct sub-conditional sequence base for the current conditional sequence based on the conditional frequent events.
5. *Recursive Mining for Sub-Conditional Sequence Base.* Recursively repeat steps 2, 3 and 4, until no more sub-conditional sequence base need to be constructed.

Figure 3.3: Overview of the proposed CSB-mine algorithm.

3.3.1 Preprocessing

The first step in the CSB-mine algorithm is to construct the initial conditional sequence base from the web access sequence database. The *initial conditional sequence base* and *conditional sequence base* are defined as follows.

Definition 3.1. The *initial conditional sequence base*, denoted as *Init-CSB*, is the set of all web access sequences in the given database.

Definition 3.2. The *conditional sequence base* of an event e_i based on prefix sequence S_{prefix} , denoted as $CSB(S_c)$, where $S_c = S_{prefix} + e_i$, is the set of all long suffix sequences of e_i in sequences in $CSB(S_{prefix})$. $CSB(S_{prefix}) = Init-CSB$, if $S_{prefix} = \emptyset$.

We also call $CSB(S_c)$ the conditional sequence base of conditional prefix S_c . The initial conditional sequence base can also be denoted as $CSB(\emptyset)$, with $S_c = \emptyset$.

3.3.2 Constructing Event Queues for Conditional Sequence Base

The second step of the CSB-mine algorithm is to construct event queues for $CSB(S_c)$ (for *Init-CSB*, $S_c = \emptyset$). The process performs the following four steps: (1) finding conditional frequent events from $CSB(S_c)$; (2) creating a Header Table; (3) constructing e_i -queue for each conditional frequent event; and (4) deleting non-frequent events. The *conditional frequent event* is defined as follows.

Definition 3.3. The *conditional frequent event* is the event whose support in the given conditional sequence base is not less than the support threshold, $MinSup$.

To find conditional frequent events in $CSB(S_c)$, we need to identify those events with support greater than or equal to $MinSup$. This is given in the following equation, where

Algorithm 3.1 Construct_EQ($MinSup$, $CSB(S_c)$, E)**Input:**

$MinSup$ - the minimum support threshold
 $CSB(S_c)$ - the conditional sequence base of S_c
 $E = \{e_i\}$ - the set of all unique access events

Output:

$CSB(S_c)$ with Header Table HT and event queues

Process:

```

1: Initialize the Header Table  $HT \leftarrow \{\emptyset\}$ 
2: for all  $e_i \in E$  do
3:   if  $Sup(e_i) \geq MinSup$  then
4:     Insert  $e_i$  into  $HT$ 
5:   end if
6: end for
7: for all conditional sequence  $S_i \in CSB(S_c)$  do
8:   for all  $e_i \in HT$  do
9:     Insert the first item labeled  $e_i$  in  $S_i$  into  $e_i$ -queue
10:  end for
11:  Delete all items of events  $\notin HT$  from  $S_i$ 
12: end for
13: return  $CSB(S_c)$  with  $HT$  and event queues

```

$|\{S_j | e_i \in S_j, S_j \in CSB(S_c)\}|$ is the number of sequences which contains the item labeled e_i in $CSB(S_c)$, and $|Init-CSB|$ is the length of $Init-CSB$.

$$Sup(e_i) = \frac{|\{S_j | e_i \in S_j, S_j \in CSB(S_c)\}|}{|Init-CSB|} \geq MinSup$$

Then, all the conditional frequent events form the entire Header Table of $CSB(S_c)$. A linked-list structure for each conditional frequent event e_i , called e_i -queue, is created. Each item of e_i -queue is the first item labeled e_i in sequences of $CSB(S_c)$. The head pointer of e_i -queue is recorded in the Header Table. Finally, as all the items of sequences in $CSB(S_c)$ which are labeled as non-frequent events are no longer needed, they are discarded. The **Construct_EQ** algorithm for constructing event queues for $CSB(S_c)$ is given in Algorithm 3.1.

Example: If $Init-CSB = \{abdac, eaebcac, babfae, afbacfc\}$, then the results obtained after constructing the Header Table and event queues are shown in Figure 3.4. To be qualified as a conditional frequent event (with $MinSup = 75\%$ and $|Init-CSB| = 4$), an event must have a count of at least 3. Therefore, the conditional frequent events are $(a : 4)$, $(b : 4)$ and $(c : 3)$. Each access event is denoted as $(event : count)$, where event is the event name and count is the number of sequences which contains the item labeled as event in $Init-CSB$. The

Figure 3.4: *Init-CSB* with the Header Table and event queues.

a-queue, *b*-queue and *c*-queue are shown by the dashed lines starting from the Header Table. The items labeled as non-frequent events *d*, *e* and *f* in each sequence are deleted. Similarly, for any subsequent conditional sequence base, the Header Table and event queues can also be constructed using the **Construct_EQ** algorithm.

Complexity analysis: In this step, we need two scans of the current conditional sequence base: (1) Scan $CSB(S_c)$ once, find all conditional frequent events and create a Header Table; (2) Scan $CSB(S_c)$ again, construct event queues and delete non-frequent events. Here, we assume equal cost for each scan. Thus, the total cost of this step is $O(n)$, where n is the total number of events in $CSB(S_c)$. The WAP-mine algorithm also needs to construct the event queues for the current WAP-tree. In terms of storage space, the proposed CSB-mine algorithm should use more memory than the WAP-mine algorithm in this step due to the compact structure of WAP-tree. But we will see later that the WAP-mine algorithm will use much more space than the CSB-mine algorithm in subsequence phases. In addition, the PrefixSpan algorithm also uses a similar technique, named the pseudo-projection, to reduce the cost of constructing projected databases recursively. However, the pseudo-projection needs to store both the pointers to the sequences and offsets of the suffix sequences. Therefore, compared with the event queues, the pseudo-projection in the PrefixSpan algorithm should use more memory.

3.3.3 Constructing Sub-Conditional Sequence Base

The sub-conditional sequence base is defined as follows.

Definition 3.4. $CSB(S_{prefix} + e_i)$ is called the *sub-conditional sequence base* of $CSB(S_{prefix})$, if $e_i \neq \emptyset$.

For each access event e_i in the Header Table of $CSB(S_c)$, the **Construct_Sub_CS**

Algorithm 3.2 Construct_Sub_CSB($CSB(S_c)$, e_i)**Input:** $CSB(S_c)$ - the conditional sequence base of S_c e_i - a given event in Header Table of $CSB(S_c)$ **Output:** $CSB(S_c + e_i)$ - the conditional sequence base of e_i based on $CSB(S_c)$ **Process:**

- 1: Initialize $CSB(S_c + e_i) \leftarrow \{\emptyset\}$
- 2: **for all** item in e_i -queue of $CSB(S_c)$ **do**
- 3: Insert its suffix sequence into $CSB(S_c + e_i)$
- 4: **end for**
- 5: **return** $CSB(S_c + e_i)$

Figure 3.5: Construction of $CSB(a)$ based on $Init-CSB$.

algorithm for constructing $CSB(S_c + e_i)$ based on $CSB(S_c)$ is given in Algorithm 3.2.

Example: For the $Init-CSB$ shown in Figure 3.4, we obtain all suffix sequences of a by following the a -queue as $CSB(a)$, which is one of the sub-conditional sequence base of $Init-CSB$. The result is shown in Figure 3.5. $CSB(a)$ contains $\{bac, bcac, ba, bacc\}$.

Complexity analysis: In this step, we need to follow one event queue e_i -queue and scan all related suffix sequences in the current conditional sequence base $CSB(S_c)$. Here, we assume equal cost for scanning each event in e_i -queue and $CSB(S_c)$. Thus, the total cost of this step is $O(n)$, where n is the total number of events in the e_i -queue and the suffix sequences of e_i in $CSB(S_c)$. In the WAP-mine algorithm, the sub-conditional sequence base is also constructed. This is done by constructing the conditional WAP-tree first, and then, extracting the sub-conditional sequence base from the conditional WAP-tree. The construction of the conditional WAP-tree is costly, and the conditional WAP-tree is not used in subsequent processing, if it has more than one branch. In the CSB-mine algorithm, we have avoided this disadvantage. And our method for constructing sub-conditional sequence base is

more efficient than using the conditional WAP-tree. It is because the WAP-mine algorithm needs to scan the current conditional sequence base once to construct the conditional WAP-tree, and then, construct the sub-conditional sequence base by following the event-queues of the conditional WAP-tree. But in the CSB-mine algorithm, the sub-conditional sequence base can be constructed by scanning the current conditional sequence base only once. In terms of storage space, the proposed CSB-mine algorithm should use less memory than the WAP-mine algorithm, since the latter has to store the current conditional WAP-tree and the corresponding conditional sequence base at the same time. In addition, the PrefixSpan algorithm should also use a little more time to construct its projected database (similar to the conditional sequence base) in this phase, because the position of each suffix sequence must be calculated first from the corresponding sequence pointer and offset.

3.3.4 Single Sequence Testing for Conditional Sequence Base

In this step, if all sequences in $CSB(S_c)$ can be combined into a single sequence, the mining of $CSB(S_c)$ will be stopped. This single sequence will be used to form a part of the final sequential access patterns. Otherwise, we construct *Sub-CSBs* for $CSB(S_c)$ and perform recursive mining. The **Test-CSB** algorithm for testing whether all sequences in $CSB(S_c)$ can be combined into a single sequence is given in Algorithm 3.3.

Example: For $CSB(a) = \{bac, bcac, ba, bacc\}$, the first item of each sequence can be combined into one item ($b : 4$), but the second item cannot. The combination is stopped and returns the *failed* flag. For $CSB(aa) = \{c, c, cc\}$, the sequences can be combined into a single sequence $c : 3$ and the *successful* flag is returned.

Complexity analysis: In this step, we need to scan the current conditional sequence base $CSB(S_c)$ once, if all the sequences can be combined into a single sequence. Otherwise, we only need to scan a part of $CSB(S_c)$. Thus, for the worst cases, the total cost of this step is $O(n)$, where n is the total number of events in $CSB(S_c)$. In the WAP-mine algorithm, the conditional WAP-tree must be constructed recursively until there is only one branch in the tree. In fact, the construction of conditional WAP-trees can be avoided, because what we want to know from this process is only on whether all sequences in the given conditional sequence base can be combined into a single sequence or not. Therefore, the single sequence testing method, **Test-CSB**, can be used instead of constructing conditional WAP-trees. The Single Sequence Testing process will be terminated when all items in the given conditional sequence base can be merged into the existing single sequence. With such simple testing method,

Algorithm 3.3 Test_CSB($CSB(S_c)$, HT)**Input:** $CSB(S_c)$ - the conditional sequence base of S_c HT - the Header Table of $CSB(S_c)$ **Output:**test result - *successful* or *failed* flag $SingleSeq$ - the single sequence of $CSB(S_c)$ **Process:**

```

1: Initialize  $SingleSeq \leftarrow \emptyset$ 
2: if  $CSB(S_c) = \emptyset$  then
3:    $SingleSeq \leftarrow \emptyset$ 
4:   return successful and  $SingleSeq$ 
5: end if
6: for  $i = 1$  to  $n$ , which is the maximum length of all sequences  $S_i \in CSB(S_c)$  do
7:   if all the  $i$ th items in each sequence  $S_i \in CSB(S_c)$  are the same event  $e$  and the total
   count of these items  $\geq MinSup \times |Init-CSB|$  then
8:     Create a new item  $e$  with the count and insert it into  $SingleSeq$ 
9:   else
10:     $SingleSeq \leftarrow \emptyset$ 
11:    return failed and  $SingleSeq$ 
12:   end if
13: end for
14: return successful and  $SingleSeq$ 

```

the efficiency of the mining process can be greatly improved. In terms of storage space, the proposed CSB-mine algorithm should use similar memory compared with the WAP-mine algorithm, because we only need to store the current conditional sequence base and a single sequence, but WAP-mine has to store the current WAP-tree and a new conditional WAP-tree at the same time. In the PrefixSpan algorithm, the processing of one projected database will be terminated, when no frequent subsequences can be generated. However, for the projected database with only the same sequences (i.e., all sequences can be combined into a single sequence), it still needs further projections till the generated projected database is empty. Such processes are costly and redundant, especially for long sequential access patterns. Our single sequence testing method can avoid constructing redundant conditional sequence bases effectively in such cases.

3.3.5 The Complete CSB-mine Algorithm

The complete **CSB-mine** algorithm for mining sequential access patterns from a given web access sequence database is shown in Algorithm 3.4.

Example: The complete sequential access patterns with $MinSup = 75\%$ is shown in

Algorithm 3.4 CSB-mine($MinSup$, WAS_{DB})**Input:** $MinSup$ - the minimum support threshold $WAS_{DB} = \{S_i | 1 \leq i \leq m\}$ - the web access sequence database $E = \{e_i | 1 \leq i \leq n\}$ - all unique access events in WAS_{DB} **Output:** SAP - the set of all sequential access patterns**Process:**

- 1: Initialize $SAP \leftarrow \{\emptyset\}$
- 2: Preprocessing to construct $Init-CSB$, i.e., $CSB(S_c)$ with $S_c = \emptyset$
- 3: Use **Construct_EQ** to construct event queues for $CSB(S_c)$
- 4: Use **Test-CSB** to test single sequence for $CSB(S_c)$
- 5: **if** the test is *successful* **then**
- 6: Frequent sequence $FS \leftarrow S_c + SingleSeq$
- 7: Insert all ordered combinations of items in FS into SAP
- 8: **else**
- 9: **for all** e_j in Header Table of $CSB(S_c)$ **do**
- 10: Use **Construct_Sub-CSB** to construct $CSB(S_c + e_j)$
- 11: Set $S_c = S_c + e_j$ and recursively mine $CSB(S_c)$ from Line 3
- 12: **end for**
- 13: **end if**
- 14: **return** SAP

Table 3.2: The sequential access patterns of the sample database.

Length of Patterns	Sequential Access Patterns
1	$a : 4, b : 4, c : 3$
2	$aa : 4, ab : 4, ac : 3, ba : 4, bc : 3$
3	$aac : 3, aba : 4, abc : 3, bac : 3$
4	$abac : 3$

Table 3.2.

As can be seen from the proposed mining algorithm, CSB-mine, it has significant advantages when compared with the original WAP-mine algorithm. Firstly, it avoids the costly construction of the initial WAP-tree and the re-construction of the conditional WAP-trees. Secondly, the construction method for the sub-conditional sequence base is more efficient than the method based on the conditional WAP-trees. These two special features of the CSB-mine algorithm help improve the efficiency of the mining process significantly. And this will be shown in the next section on performance evaluation. In terms of storage space, the proposed CSB-mine algorithm should use similar memory compared with the WAP-mine algorithm. However, memory usage is not the main objective for proposing the CSB-mine algorithm.

3.4 Performance Evaluation

In this section, we evaluate the performance of the proposed CSB-mine algorithm.

3.4.1 Datasets

In this research, we have used two datasets for evaluating the performance of all proposed approaches.

The first dataset (Dataset_1) is obtained from the Microsoft Anonymous Web Data [HB99], which contains the web server logs of www.microsoft.com. It records the topic areas each user has visited within a week in February 1998. This dataset consists of two parts: Dataset_1_Training and Dataset_1_Testing. Dataset_1_Training contains a total of 32,711 web access sequences with each sequence containing from 1 up to 35 page references from a total of 294 topics. Dataset_1_Testing contains a total of 5,000 web access sequences.

The second dataset (Dataset_2) is obtained from the web server logs of a web forum, called “North Latitude One BBS” (<http://bbs.nlonet.net>), for the period from 01-May-2005 to 31-May-2005. This dataset contains a total of 256,683 access records. A total of 6,828 web access sequences is derived from the access records, with each sequence containing from 1 up to 140 records. This web forum consists of a total of 64 topics including 7 main topics (such as Past_time, Sports, Computer, etc.) and 57 sub-topics (such as MovieTV in Past_time, Soccer in Sports, etc.). The web forum is mainly used by graduate students from Nanyang Technological University, Singapore.

3.4.2 Experiments

We evaluate the performance of the proposed CSB-mine algorithm and compare it with the WAP-mine [PHMAZ00] algorithm and the PrefixSpan [PHMA⁺01] algorithm for mining sequential access patterns. The WAP-mine algorithm is one of the most efficient algorithms that mine common sequential access patterns from a highly compressed data structure known as WAP-tree. As evaluated in [PHMAZ00], the performance of the WAP-mine algorithm is an order of magnitude faster than other Apriori-based algorithms. The PrefixSpan algorithm was proposed by the same authors of WAP-mine. The corresponding performance study [PHMA⁺01] has shown that the PrefixSpan algorithm runs considerably faster than the Apriori-based algorithms and another recently proposed FreeSpan [HPMA⁺00] algorithm. Therefore, we only compare the CSB-mine algorithm with the WAP-mine and PrefixSpan algorithms.

We have implemented the three algorithms, i.e., CSB-mine, WAP-mine and PrefixSpan, in C++. All experiments are performed on a 3.4GHz Intel Pentium 4 PC machine with 1.0GB memory, running on Microsoft Windows XP Professional. Both Dataset_1 and Dataset_2 are used for evaluating the two algorithms. Note that only 22,716 web access sequences of the Dataset_1_Training and 5,175 web access sequences of the Dataset_2 which have more than one access event are used for the evaluation.

We have conducted two experiments to evaluate the performance of the proposed algorithm. The first experiment evaluates the efficiency of the three algorithms with respect to different support thresholds. In the experiment, we measure the runtime performance using different support thresholds. For Dataset_1_Training, we use support thresholds varying from 0.1% to 10.0%. For Dataset_2, we use support thresholds varying from 5.0% to 60.0%. The experimental results are given in Figure 3.6(a) and Figure 3.6(b) which have shown that the runtime of the WAP-mine algorithm increases more sharply than that of both the CSB-mine and PrefixSpan algorithms, when the support threshold decreases. In addition, the CSB-mine algorithm always requires less runtime than both the WAP-mine and PrefixSpan algorithms. The CSB-mine and PrefixSpan algorithms have similar efficiency. However, the CSB-mine algorithm runs much faster than the PrefixSpan algorithm when the support threshold is small. It is because the PrefixSpan algorithm needs to construct many redundant projected databases for long sequential access patterns with small support values. When a larger support threshold is used, the runtimes of these three algorithms will get very close to each other.

The second experiment evaluates the scalability of the three algorithms with respect to different sizes of Dataset_1_Training and Dataset_2. In the experiment, we measure the runtime performance using a fixed support threshold with different numbers of web access sequences. For Dataset_1_Training, we use $MinSup = 0.1\%$ with the number of web access sequences varying from 4,000 to 22,716. For Dataset_2, we use $MinSup = 20.0\%$ with the number of web access sequences varying from 1,000 to 5,175. The experimental results are given in Figure 3.7(a) and Figure 3.7(b) which have shown that the CSB-mine algorithm has achieved better scalability than both the WAP-mine and PrefixSpan algorithms especially for large datasets.

Figure 3.6: Performance results of CSB-mine vs. WAP-mine vs. PrefixSpan based on different support thresholds.

Figure 3.7: Performance results of CSB-mine vs. WAP-mine vs. PrefixSpan based on different numbers of sequences.

3.5 Summary

In this chapter, we have proposed an efficient algorithm, called CSB-mine, for mining sequential access patterns from web access sequence databases. The performance of the proposed CSB-mine algorithm has been evaluated in comparison with the WAP-mine and PrefixSpan algorithms in terms of efficiency and scalability. The experimental results have shown that the CSB-mine algorithm performs much more efficient than both the WAP-mine and PrefixSpan algorithms, especially when the support threshold becomes smaller and the number of web access sequences gets larger. In addition, the CSB-mine algorithm has also achieved better scalability than both the WAP-mine and PrefixSpan algorithms.

In addition, we have also incorporated the CSB-mine algorithm into a web recommender system to mine sequential access patterns from web usage logs for efficient and effective web recommendations. This will be discussed in the next chapter.

Chapter 4

Sequential Web Access Based Recommender System

As mentioned earlier, recommendation of web resources is one of the most commonly offered Web personalization functions, and is supported by a number of practical systems [LAR02, MGPL04, ZHC04]. The goal of web recommendation is to determine which web pages are more likely to be accessed next or in the near future by the current user. In this chapter, we apply the CSB-mine algorithm proposed in Chapter 3 into a web recommender system, called *SWARS* (Sequential Web Access based Recommender System), which provides personalized web services for accessing related web pages efficiently and effectively. In traditional approaches [MDLN02, GH03], the recommendation engine needs to load all discovered sequential access patterns as input and generates a set of recommended resources by matching the user's current user access sequence against the discovered patterns one by one. Such task is usually very costly in terms of runtime and storage. In the proposed system, a highly compacted data model, named Pattern-tree, is constructed based on the discovered sequential access patterns. The Pattern-tree facilitates a more efficient strategy for matching user access patterns and generating recommendation rules. Different evaluation measures including precision, satisfaction and applicability are also proposed to measure the performance of the *SWARS* recommender system.

The rest of this chapter is organized as follows. In Section 4.1, we review research work on web personalization and recommendation. Then, we introduce the architecture of the proposed web recommender system in Section 4.2. The algorithms for constructing the recommendation model, i.e., Pattern-tree, and generating recommendation rules are then

presented in Sections 4.3 and 4.4, respectively. The performance of the proposed web recommender system is evaluated in Section 4.5. Finally, a summary of this chapter is given in Section 4.6.

4.1 Web Personalization and Recommendation

Web personalization [EV03, BFR⁺03, PPS03] is the process of customizing the content and structure of a website to the needs of a user by taking advantage of the knowledge acquired from the analysis of the user's access behavior. Recommendation of web resources is one of the most commonly offered Web personalization functions. For the past few years, various statistical and knowledge discovery techniques have been proposed and applied to web personalization and recommendation. These techniques can be classified into rule-based filtering, content-based filtering, collaborative filtering and web usage mining approaches [EV03] .

4.1.1 Rule-based Filtering Approaches

In *rule-based filtering* approaches, users usually are required to answer a list of questions during their registration procedure. According to users' answers, a number of rules are generated as the static user models, which are used to determine how to tailor the Web content to the needs of users. Rule-based filtering approaches are popularly used in online e-commerce websites, such as Amazon.com and Dell.com. However, all rule-based filtering approaches suffer from the same problem, i.e., they usually require users' involvement, which is quite troublesome and inconvenient for users using such systems.

4.1.2 Content-based Filtering Approaches

Content-based filtering approaches apply machine learning techniques to analyze web content accessed by a user in order to recommend web pages that are similar to the user's past preference. Letizia [Lie95] is a client side agent that monitors the user's browsing behavior and searches pages that are related to the user's interest for recommendations. The main problem of content-based filtering approaches is the difficulty of analyzing the Web content and measuring the semantic similarity between Web pages.

4.1.3 Collaborative Filtering Approaches

One of the most successful and widely used web personalization techniques is *collaborative filtering*. Collaborative filtering works by building a database of user item preferences. A current user is matched against the database to discover similar “neighbors”, which refers to other users with historically similar tastes as the current user. Items which the neighbors like are then recommended to the current user, as he/she will probably also like them. GroupLens [KMM⁺97] is a system that uses a purely collaborative filtering approach to make recommendations of Usenet news, so as to help people find desirable articles from a huge stream of news feed. However, such approaches are quite inefficient especially for large websites containing lots of pages. Since pure collaborative filtering approaches can be restrictive, some personalization approaches have proposed a hybrid approach that combines content-based filtering approach with the collaborative filtering approach. For example, in WebWatcher [JFM97], it is a web tour guide software agent that accompanies a user from page to page by recommending appropriate hyperlinks based on the content of the web pages the user visits and a partial understanding of each user’s interests. Yoda [SKCM01] is a web-based recommendation system, which combines collaborative filtering with content-based querying to achieve good accuracy and scalability in real-time. It is designed as an adaptive model to be trained off-line and later deployed for real-time on-line recommendation. Miller *et al.* [MKR04] presented the PocketLens collaborative filtering algorithm along with five peer-to-peer architectures for supporting personalized recommender system. The PocketLens algorithm divides the recommendation process into two parts. At first, it searches the network to find neighbors for building a similarity model. And then, the similarity model is used to make recommendations to the user.

4.1.4 Web Usage Mining Approaches

Recently, a number of web personalization applications have adopted web usage mining techniques [PPPS03], which mine the web logs for user models and personalization. The proposed web usage mining techniques include association rule mining, sequential pattern mining and clustering.

Surflen [FBH00] is an information recommendation system using association rules mined from the navigation history of users to suggest interesting web pages to users. The experimental results show that the proposed approach can achieve good recommendations. However, the authors did not give a performance comparison between the proposed system with other

collaborative recommendation systems. Mobasher *et al.* [MDLN01] proposed an effective and scalable technique for web personalization based on association rules discovered from web usage data. A Frequent Itemsets Graph is constructed for storing the discovered frequent itemsets. And then, it is utilized to produce recommendations efficiently in real-time. The experimental results show that the proposed approach can achieve better overall recommendation effectiveness than traditional collaborative filtering based techniques, such as k -Nearest-Neighbor (k NN) approach. Lin *et al.* [LAR02] proposed a new collaborative recommendation approach based on adaptive-support association rules. Both association rules between users and articles are used for making recommendation. Experimental results show that the proposed approach performs much better recommendation than traditional collaborative recommendation approaches. Kazienko [Kaz04] proposed an approach to combine association rules within and between web user sessions into complex association rules to compute ranking lists of web pages for recommendation. All recommendation tasks are distributed between many agents that communicate their knowledge with each other. Mobasher *et al.* [MDLN02] proposed an efficient framework for web personalization based on sequential and non-sequential pattern discovered from web usage data. The results of the performance comparison indicate that less constrained patterns, such as frequent itemsets or general sequential patterns are more effective than contiguous sequential patterns (e.g., frequent navigational paths) in web personalization and recommendation systems. A similar conclusion was also given in the research work [GH03] by Gery *et al.*. Mobasher *et al.* [MCS99] proposed an effective technique for capturing common user profiles based on usage-based clustering of URLs and performing real-time personalization. Once the URL clusters have been computed, the recommendation engine matches the current active user session to the URL clusters to compute a set of recommended URLs and returns the last requested page with a set of related links to the user. Real web usage data have been used for evaluating the proposed approach.

Compared with other approaches, web usage mining techniques can potentially achieve more accurate personalization.

4.2 System Architecture

Figure 4.1 shows the architecture of the proposed *SWARS* recommender system. Firstly, all users' web access requests of a website are recorded by the Web server of the website and stored into its Web Server Logs. The *CSB-mine Algorithm* component is then applied to mine sequential access patterns from the Web Server Logs. The *Pattern-tree Construction*

Figure 4.1: The architecture of the proposed web recommender system.

component constructs the recommendation model, called *Pattern-tree*, from the mined sequential access patterns. Both the CSB-mine algorithm and the Pattern-tree construction processes are carried out off-line. When a user visits the website, the user's HTTP requests in the current browsing session are recorded in order, and the current access sequence is constructed. Matching the user's current access sequence from the recommendation model of the Pattern-tree, the *Recommendation Rules Generation* component will generate recommendation rules. The recommended links will then be inserted into the current requested page dynamically.

An example browser display is shown in Figure 4.2. The upper frame shows the original requested web page and the lower frame displays a list of recommended links.

4.3 Pattern-tree Construction

A Sequential Access Pattern Tree (or Pattern-tree) model is proposed for storing sequential access patterns compactly and searching the best matching pattern of a given web access sequence efficiently. The Pattern-tree is based on the Trie [Knu98] data structure, which is a tree-based data structure for storing strings in order to facilitate fast pattern matching. In general, the set of sequential access patterns can be treated as a set of strings over a finite alphabet E (the set of unique requested resources). Every Pattern-tree node is labeled with a symbol (requested resource) from E with a corresponding support value. Once the Pattern-tree is constructed, the original sequential access patterns are no longer required for subsequent processing. In addition, if an incremental sequential pattern mining technique

Figure 4.2: An example browser display of the requested web page with recommended links.

[ZKY02] were used, the Pattern-tree would be easily updated incrementally.

To construct a Pattern-tree, only one scan of all sequential access patterns is needed. The **Pattern-tree_Construction** algorithm based on the set of discovered sequential access patterns is given in Algorithm 4.1.

Example: The Pattern-tree of the discovered sequential access patterns given in Table 3.2 is shown in Figure 4.3. It is constructed as follows. Starting from the virtual root node R , insert the sequential access patterns $a : 4$, $b : 4$, and $c : 3$ into the Pattern-tree, each of which becomes a new child node of R , labeled as (resource : support). Next, insert the sequential access patterns $aa : 4$, $ab : 4$, $ac : 3$, $ba : 4$, $bc : 3$ into the Pattern-tree. Then, five branches “ $(a : 4) \rightarrow (a : 4)$ ”, “ $(a : 4) \rightarrow (b : 4)$ ”, “ $(a : 4) \rightarrow (c : 3)$ ”, “ $(b : 4) \rightarrow (a : 4)$ ” and “ $(b : 4) \rightarrow (c : 3)$ ” are derived, in which arrows are used to point from parent nodes to child nodes. When inserting $ac : 3$, there has been a child node of R labeled as $(a : 4)$, so this node is shared and the maximum support value 4 is used as the support value. The remaining insertion process can be derived accordingly. From the Pattern-tree, all the sequential access patterns can be visited by traversing the path starting from the root node of the tree.

Complexity analysis: Inserting a sequential access pattern S into the Pattern-tree requires two steps. The first step searches for a matching prefix sequence S_{prefix} of S in the Pattern-tree, and the second step creates a new path for the remaining non-matching suffix sequence S_{suffix} of S in the Pattern-tree. Here, we assume equal costs for matching an

Algorithm 4.1 Pattern-tree_Construction(SAP)**Input:** $SAP = \{S_i\}$ - the set of all discovered sequential access patterns**Output:** $Root$ - the root node of the Pattern-tree based on SAP **Process:**

- 1: Create an empty node $Root$ as the root node of the Pattern-tree
- 2: **for all** sequential access patterns $S_i \in SAP$, denoted as $S_i = e_1e_2 \cdots e_n$ **do**
- 3: Set $Root$ as $current_node$
- 4: **for** $i = 1$ **to** n **do**
- 5: **if** $current_node$ has a child node $child_node$ labeled as e_i **then**
- 6: Update the support value of $child_node$ to the maximum support value between S_i and $child_node$
- 7: **else**
- 8: Create a new node $child_node$ as a child of $current_node$ labeled as e_i with the support value of S_i
- 9: **end if**
- 10: Set $child_node$ as $current_node$
- 11: **end for**
- 12: **end for**
- 13: **return** $Root$

Figure 4.3: The Pattern-tree derived from the sequential access patterns given in Table 3.2.

existing node and creating a new node. Thus, the total cost of inserting a sequential access pattern S with length m is $O(|S_{prefix}|) + O(|S_{suffix}|) = O(|S|) = O(m)$. The complexity of constructing the entire Pattern-tree from scratch is equivalent to the cumulative cost of inserting all sequential access patterns $S_1, S_2, \dots, S_l$ with a total length n , which is $O(|S_1|) + O(|S_2|) + \dots + O(|S_l|) = O(n)$.

In terms of memory usage, the generated Pattern-tree can be kept in persistent storage, with frequently accessed patterns cached in memory. Moreover, the Pattern-tree uses significantly less space than the mined sequential access patterns because of its compact Trie structure. For example, we only need a Pattern-tree with 14 nodes for storing a total of 29 items in the simple sequential access patterns given in Table 3.2.

Web logs are constantly updated in real-life. However, for web recommendation, we can assume it to be relatively static for a short period of time. Since the construction of the Pattern-tree is an off-line process, we can execute it anytime as necessary. The frequency for updating the Pattern-tree varies depending on the volatility of the website. For large websites like Amazon.com, the Pattern-tree needs to be updated fairly frequently such as daily during off-peak hours. In this research, we do not elaborate on this incremental update aspect, which is a separate research topic on its own.

4.4 Recommendation Rules Generation

The *Recommendation Rules Generation* component searches for the best-matching access path from the Pattern-tree according to the current access sequence of a user. If the best-matching path is found, all the resources represented by the child nodes of the path on the Pattern-tree will be selected and ranked in descending order according to their support values as candidates for recommendations.

Definition 4.1. The *recommendation rule* of a given current access sequence S based on the Pattern-tree PT is defined as “ $S \Rightarrow R_o(S, PT)$ ”, where $R_o(S, PT)$ is an ordered set of recommended resources.

In general, a longer current access sequence will have a lower probability for finding an exact matching path from the Pattern-tree. To increase the applicability of recommendation rules generation, the suffix sequences of the current access sequence will be considered when the exact matching path of the whole access sequence cannot be found. Consequently, the head of the current access sequence may be removed repeatedly till a matching path is found or when no more item can be removed from the current access sequence. In addition, the length of the longest path in the Pattern-tree is the depth of the Pattern-tree. The matching path will not exist when the length of the current access sequence is longer than the depth of the Pattern-tree. Therefore, the current access sequence is always trimmed by removing some earlier items to be shorter than the depth of the Pattern-tree before the sequence matching process.

Typically, recommendation rules generated from shorter matching paths usually have lower accuracy. In order to improve the precision of recommendation rules generation, only web access sequence longer than a given threshold is processed. In other words, the sequence matching process will stop if the length of the remaining access sequence is less than the

Algorithm 4.2 Recommendation_Rules_Generation(*Root*, *S*, *MinLength*, *MaxLength*)**Input:***Root* - the root node of the Pattern-tree*S* = $e_1e_2 \cdots e_n$ - the current access sequence of a user*MinLength* - the minimum valid length of the current access sequence*MaxLength* - the maximum valid length of the current access sequence**Output:** $S \Rightarrow R_o(S, PT)$ - the recommendation rule for *S* based on *PT***Process:**

```

1: Initialize  $R_o(S, PT) \leftarrow \{\emptyset\}$ 
2: if  $|S| > MaxLength$  then
3:   Remove the first  $|S| - MaxLength + 1$  items from S
4: end if
5: while  $|S| \geq MinLength$  do
6:   Set Root as current_node
7:   Initialize matching  $\leftarrow false$ 
8:   for each  $e_i$  in order in the current S do
9:     if current_node has a child node child_node labeled as  $e_i$  then
10:      Set child_node as current_node
11:      matching  $\leftarrow true$ 
12:     else
13:       Remove the first item from S
14:       matching  $\leftarrow false$ 
15:       break
16:     end if
17:   end for
18:   if matching = true and current_node has child nodes then
19:     Insert these child nodes into  $R_o(S, PT)$  ordered by their support values
20:   break
21:   end if
22: end while
23: return  $S \Rightarrow R_o(S, PT)$ 

```

threshold.

Suppose the current access sequence of a user is $S = e_1e_2 \cdots e_n$, the algorithm for generating the recommendation rule for *S*, **Recommendation_Rules_Generation**, is given in Algorithm 4.2.

Example: Let's consider the Pattern-tree shown in Figure 4.3. Suppose the current access sequence of a user is $S = cab$ (infrequent events have been removed) and the length thresholds of web access sequence are $MinLength = 2$ and $MaxLength = 4$ (the depth of the Pattern-tree in Figure 4.3 is 5). The recommendation rule for *S* is generated as follows. The sequence matching process starts from the first sequence item *c* and the child nodes of the root of the Pattern-tree. We found a node (*c* : 3) that is pointed directly

by the root, then the second sequence item a is scanned. But the node $(c : 3)$ has no child node labeled as a . The first item of S is then removed, and then $S = ab$. Because $|S| = 2 \geq MinLength$, the sequence matching process is repeated. Now, the matching path $(a : 4) \rightarrow (b : 4)$ can be found in the Pattern-tree. The node $(b : 4)$ has two children $(a : 4)$ and $(c : 3)$. Finally, the recommendation rule for the current access sequence $S = cab$ is generated as “ $cab \Rightarrow \{a, c\}$ ”, where the recommended resources are ordered by their support values. Based on the recommendation rules, the corresponding web resources can be determined and recommended. In our approach, the top 6 items in the recommended resource set are used for recommendations.

Complexity analysis: The cost of looking up the current access sequence S with length m in the Pattern-tree with $MinLength = L_{min}$ and $MaxLength = L_{max}$ is $O(\min(m, L_{max}))$. The maximum possible number of lookups is $L_{max} - L_{min}$. This gives the total cost of matching S as $O((L_{max} - L_{min}) \times \min(m, L_{max}))$. The complexity of recommendation rule generation is thus $O(L_{max}^2)$, where L_{max} is usually less than the depth of the Pattern-tree. Traditional approaches need to search the best matching pattern by matching the current access pattern against a list of sequential access patterns. Thus, the complexity is $O(\min(m, L_{max}) \times n)$ in the worst case, where n is the number of sequential access patterns. Since L_{max} is usually much less than n , it is obvious that the proposed Pattern-tree based recommendation rule generation approach is much more efficient than traditional approaches.

Generally, it is inevitable for the automatically generated recommendation rules to contain errors. As the proposed system currently does not allow users to rate each recommendation rule, it is unable to automatically correct any bad recommendation rules. Nevertheless, typical web users do not generally have time to provide feedback/rating unless it is something they feel very strongly about. In practice, we could develop an analysis mechanism to analyze and prune recommendation rules that users seldom or never follow.

4.5 Performance Evaluation

In this section, we evaluate the performance of the proposed *SWARS* recommender system.

As the Pattern-tree only stores frequent web access sequences (with a support value of at least $MinSup$), if the current access sequence does not include a frequent suffix sequence, the generated recommended resource list will be empty, i.e., no recommendations. In practise, such recommendation rule is not applicable for web recommendation. In addition, among applicable recommendation rules, some of them may recommend a correct resource (the user

want in the next step) or satisfied resource (the user want in the next few steps) to the user, others may not. Therefore, we give the definitions of applicable rule, correct rule and satisfactory rule at first.

4.5.1 Evaluation Measures

Definition 4.2. Let $S = e_1e_2 \cdots e_ke_{k+1} \cdots e_n$ ($n > MinLength$) be a web access sequence in the testing dataset. Suppose we generate a recommendation rule “ $S_{prefix} \Rightarrow R_o(S_{prefix}, PT)$ ” for a prefix sequence $S_{prefix} = e_1e_2 \cdots e_k$ ($MinLength \leq k < n$) based on the corresponding Pattern-tree PT , where the recommended resource list $R_o(S_{prefix}, PT) = \{a_1, a_2, \cdots, a_m\}$.

- If $R_o(S_{prefix}, PT) \neq \emptyset$, we call such rule an *applicable rule*.
- If $e_{k+1} \in R_o(S_{prefix}, PT)$, we call such rule a *correct rule*.
- If $\exists e_i \in R_o(S_{prefix}, PT)$ for $k+1 \leq i \leq k+m \leq n$, we call such rule a *m-step satisfactory rule*.

In order to objectively evaluate the feasibility of our proposed approach, we define the following three evaluation measures.

Definition 4.3. Let RR_{all} be a set of recommendation rules for the overall web recommendation, and RR_a be the subset of RR_{all} comprising all applicable recommendation rules. The *applicability* of the overall web recommendation is defined as

$$applicability = \frac{|RR_a|}{|RR_{all}|}.$$

Therefore, the applicability measure gives a rough idea of how often nonempty recommended resource lists will be generated. Generally, a smaller *MinSup* will result in a higher applicability, but with the expense of increased sequential pattern mining and Pattern-tree construction cost.

Definition 4.4. Let RR_c be the subset of RR_{all} comprising all correct recommendation rules. The *precision* for the overall web recommendation is defined as

$$precision = \frac{|RR_c|}{|RR_a|}.$$

The precision measures how probable a user will request the recommended resources in the next step.

Definition 4.5. Let $RR_s(m)$ be the subset of RR_{all} comprising all m -step satisfactory recommendation rules. The m -step *satisfaction* for the overall web recommendation is defined as

$$satisfaction = \frac{|RR_s(m)|}{|RR_a|}.$$

The satisfaction measure gives the precision that the recommended resources will be requested in the near future (within m steps). Clearly, the satisfaction and precision measures are the same when $m = 1$. In order to realistically evaluate the web recommender system, m has been set with an appropriate value of 5 to indicate that the recommended resources should be accessed in the *near* future. The satisfaction is a very important evaluation measure for web recommendation. Actually, the next resource requested by a user may not be the target resource that the user wants. In many cases, a user has to access some intermediate resources before reaching the target resource. Therefore, it is inappropriate if we only use the precision measure to evaluate a web recommender system.

4.5.2 Experiments

We have implemented the proposed system in C++. All experiments are performed on a 3.4GHz Intel Pentium 4 PC machine with 1.0GB memory, running on Microsoft Windows XP Professional. Both Dataset_1 and Dataset_2 are used for training and testing the proposed *SWARS* system. For Dataset_1, we use 22,716 valid web access sequences (with more than one access event) from the total of 32,711 web access sequences for Dataset_1_Training. And we use 1,327 valid web access sequences from the total of 5,000 web access sequences for Dataset_1_Testing. For Dataset_2, we use the web logs from 01-May-2005 to 20-May-2005 as Dataset_2_Training, which contains 3,770 valid web access sequences. And we use the web logs from 21-May-2005 to 31-May-2005 as Dataset_2_Testing, which contains 5,942 valid web access sequences.

We have conducted two experiments to evaluate the performance of the proposed *SWARS* system. In both experiments, we use the support thresholds of 0.04% and 1.0% for mining sequential access patterns and constructing the Pattern-trees from Dataset_1_Training and Dataset_2_Training respectively. The experimental results given in Table 4.1 show that the runtime for constructing the Pattern-tree is very short, and the Pattern-tree model has a high compression ratio (45.4% and 34.2% for the two experimental datasets). We then generate recommendation rules for both datasets based on the constructed Pattern-trees and the lists of mined sequential access patterns respectively. The proposed Pattern-tree matching

Table 4.1: Performance results of the proposed recommender system.

	Dataset_1	Dataset_2
Size of training dataset (sequences)	22,716	3,770
Support threshold	0.04%	1.0%
Runtime for Pattern-tree construction (s)	0.32	1.09
Size of Pattern-tree (nodes)	18,602	65,520
Size of pattern list (items)	40,961	191,667
Compression ratio	45.4%	34.2%
Size of testing dataset (sequences)	1,327	5,942
Runtime for Pattern-tree matching (s)	0.09	0.63
Runtime for pattern list matching (s)	1.21	5.40
Applicability	81.8%	86.9%

approach requires much less runtime than traditional pattern list approach for both datasets. In addition, we have achieved applicabilities of 81.8% and 86.9%, which give the proportions of applicable rules in all generated recommendation rules, for Dataset_1_Testing and Dataset_2_Testing respectively.

The first experiment measures the scalability of the precision and satisfaction of the recommendation rules generation with respect to different numbers of recommended resources (from 1 to 10). The experimental results are shown in Figure 4.4. When the number of recommended pages increases, the precision and satisfaction also increase. But, the increase is not significant after the number of recommended resources is more than 6. Although the precision and satisfaction can be further improved with more recommended resources (e.g., 10), a long list of recommended resources will affect the normal browsing activity. As such, we have used 6 as the default number of recommended resources for the web recommender system.

The second experiment measures the scalability of the satisfaction of web recommendations with respect to different numbers of steps (from 1 to 8). The experimental results are given in Figure 4.5. When the number of steps becomes larger, the satisfaction of web recommendations increases. However, the increase becomes insignificant when the number of steps exceeds 5. Therefore, we use a *5-step satisfaction* for evaluating the performance of the proposed web recommender system. For Dataset_1, when using a support threshold of 0.04% and recommending only the top 6 resources, the *5-step satisfaction* has achieved 88.3% accuracy. For Dataset_2, we have obtained a *5-step satisfaction* of 86.9% with a support threshold of 1.0% and recommending the top 6 resources.

Figure 4.4: Performance results of the proposed recommender system based on different numbers of recommended resources.

Figure 4.5: Performance results of the proposed recommender system based on m-step satisfaction.

In addition, the process for generating recommendation rules (sequence matching in the Pattern-tree) has also achieved good performance. It took only 0.100 second to generate recommendation rules for 1,327 web access sequences in Dataset_1_Testing (i.e., an average of 13,270 web access sequences per second), and 0.641 second for 5,942 web access sequences in Dataset_2_Testing (i.e., an average of 9270 web access sequences per second).

4.6 Summary

To provide personalized web services such as web recommendations, it is always necessary to model and predict web access behaviors of users on the Web. In this chapter, we have proposed an effective web recommender system called *SWARS* using sequential web access

patterns. In the proposed *SWARS* system, the sequential pattern mining algorithm CSB-mine is used to mine frequent sequential web access patterns. The mined patterns are then stored in the Pattern-tree, which is then used for matching and generating web links for online recommendations. In addition, we have also proposed different measures, namely applicability, precision and satisfaction, for performance evaluation of the proposed *SWARS* system. The experimental results have shown that the proposed web recommender system is very effective for recommending related web resources which are most probably accessed by users in the near future (within 5 steps).

Chapter 5

Periodic Association Access Pattern Mining

As mentioned in Chapter 2, many techniques, such as statistical techniques, association rule mining, sequential pattern mining, clustering and classification, have been investigated for web usage mining for discovering various web access patterns of users from web usage logs. Most of these techniques have focused mainly on mining common access patterns, which have occurred frequently within the entire duration of all access sessions. However, the periodic characteristics of access patterns are often neglected. In practice, many useful access patterns occur frequently only in a particular period such as every morning, but not in other periods due to users' surfing habits. Such access patterns are referred to as *periodic access patterns*, which are very important for understanding users' access behaviors and supporting effective personalized web services.

To describe periodic access patterns more semantically, we adopt real-life temporal concepts (such as morning, evening, etc.) and meaningful requested resources (such as topics or categories related to the content of the requested URLs) as periodic and resource attributes. In practice, different people may have different interpretations for real-life temporal concepts. For example, one may consider that 8:00 p.m. is *evening* time, while others may regard that as *night* time. For requested resources, the similar problem exists because it is difficult to interpret accurately the intention of a user during an access session. Fuzzy set theory [Zad65] is one of the appropriate techniques to describe such vagueness in information. For example, a periodic web access pattern may be “ $\{Evening(0.8)\} \Rightarrow \{Sports(0.8), Chat(0.6)\}$ ”, which can be interpreted as “The user has interests in resources on Sports and Chat during the period of Evening”.

Formal Concept Analysis (FCA) [GW97, GSW04] is a formal data analysis technique based on the ordered lattice theory. It defines formal contexts to represent relationships between objects and attributes in a domain and interprets the corresponding concept lattice to represent a lattice-based concept hierarchy. FCA has been used for knowledge discovery [Stu02] and information retrieval [KC01]. However, the traditional FCA cannot handle the fuzzy concepts, i.e., real-life temporal concepts and requested resources, described earlier. Therefore, we cannot use FCA directly for our purpose. In this research, we incorporate fuzzy set theory into FCA for constructing user behavior models from web usage logs.

In this chapter, we propose a web usage mining approach for discovering a specific kind of periodic web access patterns of individual users. The proposed approach first uses fuzzy logic to represent both periodic and resource attributes, and then incorporates them into Formal Concept Analysis (FCA) for constructing a novel user behavior model, called Personal Web Usage Lattice, which represents periodic web access activities and hierarchical relationships among them. The generated Personal Web Usage Lattice can then be used for extracting periodic web access patterns.

The rest of this chapter is organized as follows. In Sections 5.1, 5.2 and 5.3, we review the related work on periodic pattern mining, fuzzy set theory and Formal Concept Analysis. Section 5.4 presents the approach for constructing the Personal Web Usage Lattice from web usage logs and extracting periodic web access patterns from it. Section 5.5 discusses an approach for visualizing periodic association access patterns. Finally, a summary is given in Section 5.6.

5.1 Periodic Pattern Mining Approaches

Periodic patterns can be defined as recurring patterns that have temporal regularities in time-series databases. Discovering periodic patterns from time series databases is an important data mining task for many applications. According to the type of patterns, periodic patterns can be divided into *periodic association rules* and *periodic sequential patterns*. In addition, according to the characteristic of periods in the discovered patterns, periodic patterns can also be grouped into *full periodic patterns* and *partial periodic patterns*. In mining full periodic patterns, every point in time contributes (precisely or approximately) to the cyclic behavior of the time series. For example, all days in a year approximately contribute to the seasonal cycle of the year. Partial periodic patterns specify the periodic behavior of the time series at some but not all of the points in time. For example, Tom reads newspaper from 7:00

a.m. to 7:30 a.m. every weekday morning, but his activities at other times do not have much regularity. Partial periodic patterns are a looser form of periodicity than full periodic patterns.

5.1.1 Mining Periodic Association Rules

Periodic association rules are rules that associate with a set of events that occur periodically. Such association rules hold only during certain time intervals but not during others. For example, “Eggs and coffee may be ordered together primarily between 7:00 a.m. - 11:00 a.m.” and “If afternoon tea is well received between 3:00 p.m. - 5:00 p.m., dinner will sell well between 7:00 p.m. - 9:00 p.m. on weekends”. Calendar information, such as hour, day, week, month and year, is crucial in describing time intervals. Discovering association rules with such temporal intervals together with calendar information that hold during the specified time intervals may lead to useful knowledge.

Ozden *et al.* [ORS98] studied the problem of discovering *cyclic association rules* that display regular cyclic variation over time. The main idea is to divide the data into disjoint segments using time intervals specified by the user, like month, week, day, etc., and then, generate cyclic association rules that have the confidence and support above the minimum thresholds within certain time intervals. By exploiting the relationship between cycles and association rule mining, three optimization techniques (i.e., *cycle-pruning*, *cycle-skipping* and *cycle-elimination*) and two new algorithms (i.e., *sequential algorithm* and *interleaved algorithm*) were proposed for finding such rules efficiently. However, this work is unable to describe real-life time concepts and deal with multiple granularities of time, such as *the morning of every weekend*.

In [RMS98], Ramaswamy *et al.* proposed *calendric association rules* that extend *cyclic association rules* for handling multiple units of time. *Calendar algebra* was also introduced to define and manipulate groups of time intervals, such that the complicated and real-life temporal patterns can be described succinctly using simple algebraic expressions. An approach for finding fuzzy patterns in association rules was also proposed for finding patterns in the data that approximately match with user-defined patterns. However, this approach requires user’s prior knowledge about the exact temporal patterns needed to be discovered, in order to define the calendar algebraic expression.

Calendar schemas proposed in [LNWJ01, LNWJ03] enable the discovery of temporal

association rules. For example, given a calendar schema $(year, month, day)$, a calendar-based pattern within this schema can be denoted as $(*, 3, 15)$, which represents the set of time intervals corresponding to 15th of March for each year. By using calendar schema as a framework for mining temporal patterns, this work requires less prior knowledge of data than the previous approaches [ORS98, RMS98]. The well-known Apriori algorithm was extended to discover calendar-based temporal association rules. In addition, two optimization techniques were developed to improve the performance of the mining process.

5.1.2 Mining Periodic Sequential Patterns

Periodic sequential pattern mining can be viewed as an extension of sequential pattern mining by taking into account the periodical characteristic in time series data. Obviously, traditional sequential pattern mining techniques can be applied to find full periodic sequential patterns. Recently, more research focuses on mining partial periodic sequential patterns, since they occur more commonly than full periodic sequential patterns in real-world applications.

Han *et al.* [HDY99] presented several algorithms for efficient mining of partial periodic (sequential) patterns for a single period as well as multiple periods. The proposed algorithms explore some interesting properties related to partial periodicity including the Apriori property and the max-subpattern hit set property. A novel tree structure, called max-subpattern tree, was proposed to facilitate the counting of candidate patterns. It has achieved better performance than the Apriori-based algorithms. However, it requires that the periods are given in advance, which limits its applicability.

Yang *et al.* [YWY00] proposed a flexible model of asynchronous periodic patterns that may be present only within a subsequence and whose occurrences may be shifted due to disturbance. Two criteria are employed to qualify valid patterns. The main idea is that a pattern needs to repeat itself at least for a certain number of times to demonstrate its significance and periodicity. In addition, the disturbance between two *valid segments* has to be less than a reasonable bound. A two-step algorithm is devised to generate potential periods by distance-based pruning, followed by an iterative procedure to derive and validate candidate patterns, and locate the longest valid subsequence.

5.2 Fuzzy Set Theory

In this section, we review some fundamental concepts of fuzzy set theory [Zad65], which are useful in this research. The fuzzy set theory was first introduced by Zadeh in 1965 to

represent vagueness in linguistics (e.g., tall, large, heavy and young).

Definition 5.1. A *fuzzy set* A on a domain U is defined as $A = \{x, \mu_A(x) \mid x \in U\}$, where $\mu_A(\cdot) : U \rightarrow [0, 1]$ is a membership function from U to $[0, 1]$, i.e., each item x in A has a membership value given by $\mu_A(x)$.

For example, a fuzzy set A that represents real numbers “close to 5” can be denoted as $A = \{x, \mu_A(x) \mid x \in U\}$, where $\mu_A(x) = \frac{1}{1+(x-5)^2}$.

Definition 5.2. Let A be a fuzzy set on the domain U . The *cardinality* of A is defined as $|A| = \sum_{x \in U} \mu_A(x)$, where $\mu_A(x)$ is the membership value of x in A .

For example, a fuzzy set $A = \{3:0.5, 4:0.8, 5:1.0, 6:1.0, 7:0.8, 8:0.5\}$ represents the concept of “several”, where each element is denoted in the form of $x:\mu_A(x)$. Then, $|A| = \sum_{x \in U} \mu_A(x) = 0.5 + 0.8 + 1 + 1 + 0.8 + 0.5 = 4.6$.

Definition 5.3. The *intersection* of fuzzy sets A and B , denoted as $A \cap B$ is defined by $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$.

Definition 5.4. The *union* of fuzzy sets A and B , denoted as $A \cup B$ is defined by $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$.

Definition 5.5. The *similarity* between two fuzzy sets A and B is defined as

$$E(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

For example, for two fuzzy sets $A = \{30:0.1, 40:0.5, 50:0.8, 60:1\}$ and $B = \{10:1, 20:1, 30:0.9, 40:0.7\}$, $A \cap B = \{30:0.1, 40:0.5\}$, $A \cup B = \{10:1, 20:1, 30:0.9, 40:0.7, 50:0.8, 60:1\}$ and $E(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{0.1+0.5}{1+1+0.9+0.7+0.8+1} = \frac{0.6}{5.4} = 0.11$.

Definition 5.6. If A and B are two fuzzy sets on the same domain U , then A is a *subset* of B , denoted as $A \subseteq B$, if and only if $\mu_A(x) \leq \mu_B(x)$, $\forall x \in U$, where $\mu_A(x)$ and $\mu_B(x)$ are membership values of x in A and B respectively.

For example, for two fuzzy sets $A = \{30:0.1, 40:0.5\}$ and $B = \{20:1, 30:0.9, 40:0.7\}$, $A \subseteq B$.

Definition 5.7. A *fuzzy relation* on a domain $X \times Y$ is defined as $R(X \times Y) = \{(x, y), \mu_R(x, y) \mid (x, y) \in X \times Y\}$.

For example, let $X = \{\text{NYC}(\text{New York}), \text{TKO}(\text{Tokyo})\}$, $Y = \{\text{TPE}(\text{Taipei}), \text{HKG}(\text{Hong Kong}), \text{PEK}(\text{Peking})\}$ and $R(X \times Y)$ represents the relation “very close” between two cities, where

$$R(X \times Y) = \begin{matrix} & \begin{matrix} TPE & HKG & PEK \end{matrix} \\ \begin{matrix} NYC \\ TKO \end{matrix} & \begin{bmatrix} 0.3 & 0.1 & 0.1 \\ 1 & 0.7 & 0.8 \end{bmatrix} \end{matrix}.$$

5.3 Formal Concept Analysis

Formal Concept Analysis (FCA) [GW97, GSW04] is a formal data analysis technique based on the ordered lattice theory. FCA has been used for knowledge discovery [SWW98, HSWW00, Stu02, VMG04] and information retrieval [KC01, PHWZ05, CR05]. This section reviews some basic concepts of FCA and its applications.

5.3.1 Formal Context and Concept Lattice

Firstly, we review the basic concepts on formal context and concept lattice in Formal Concept Analysis [GW97].

Definition 5.8. A *formal context* is a triple $K = (G, M, I)$, where G is a set of objects, M is a set of attributes, and $I \subseteq G \times M$ is a binary relation between G and M . An object g having a relation with an attribute m is denoted as $(g, m) \in I$ and read as “the object g has the attribute m ”.

A formal context can be represented by a cross table with rows labeled by the object names and columns labeled by the attribute names. A cross in row g and column m means that the object g has the attribute m .

In Table 5.1, the formal context has three objects representing three documents, namely D1, D2 and D3, and three attributes representing three research topics, namely “Web Mining”, “Web Usage Mining” and “Formal Concept Analysis”. The symbol “ $\times$ ” is used to indicate that the object has the corresponding attribute. For example, document D1 has the attribute “Web Mining”, which implies that the document D1 belongs to the research topic “Web Mining”.

Definition 5.9. Given a formal context $K = (G, M, I)$, we define the set of attributes common to objects in $A \subseteq G$ as $A^* = \{m \in M \mid \forall g \in A : (g, m) \in I\}$, and the set of objects which have all attributes in $B \subseteq M$ as $B^* = \{g \in G \mid \forall m \in B : (g, m) \in I\}$.

	Web Mining	Web Usage Mining	Formal Concept Analysis
D1	×		
D2	×	×	
D3			×

Table 5.1: A cross table of a formal context.

Definition 5.10. A *formal concept* of a given formal context $K = (G, M, I)$ is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A^* = B$ and $B^* = A$. The sets A and B are called the *extent* and *intent* of the formal concept (A, B) respectively.

Definition 5.11. For two formal concepts (A_i, B_i) and (A_j, B_j) , (A_i, B_i) is a *sub-concept* of (A_j, B_j) , denoted as $(A_i, B_i) <_C (A_j, B_j)$, if and only if $A_i \subset A_j$ ($\Leftrightarrow B_j \subset B_i$). Equivalently, (A_j, B_j) is a *super-concept* of (A_i, B_i) . The relation $<_C$ is called the *concept relationship* (or *order*). In particular, if $(A_i, B_i) <_C (A_j, B_j)$ and there is no (A_k, B_k) such that $(A_i, B_i) <_C (A_k, B_k) <_C (A_j, B_j)$, then (A_i, B_i) is a *direct sub-concept* of (A_j, B_j) , and (A_j, B_j) is a *direct super-concept* of (A_i, B_i) . We denote this as $(A_i, B_i) \prec_C (A_j, B_j)$. $\prec_C$ is called a *direct concept relationship*. The set of all formal concepts of the given formal context $K = (G, M, I)$, denoted as C , ordered in this way is called the *concept lattice*, which is denoted as $L_C = (C, <_C)$.

The concept lattice of the formal context given in Table 5.1 is shown in Figure 5.1. This concept lattice generated from the given formal context is a complete lattice, with one concept as its lower bound called *infinimum* and one concept as its upper bound called *supremum*. Figure 5.1 also shows the intent (given at the left of each node) and the extent (given at the right of each node) of every concept. For example, the intent and extent of the concept at the top left node are $\{\text{Web Mining}\}$ and $\{\text{D1, D2}\}$ respectively.

5.3.2 Applications of FCA

Formal Concept Analysis can be used efficiently for computing frequent patterns [Stu02], and significantly reducing the number of association rules generated without losing any quality compared with that of using traditional Apriori-based algorithms [AS94]. In addition, association rules can also be visualized directly from the concept lattice. In [STB⁺01], it proposed an approach for computing bases of association rules, from which all rules can be derived. The approach consists of two steps. In the first step, an iceberg concept lattice is computed. It consists of all FCA concepts that satisfy with the user-defined minimum support. The

Figure 5.1: The concept lattice of the formal context given in Table 5.1.

TITANIC algorithm [STB⁺02] was proposed for computing the iceberg concept lattice. In the second step, the bases for the association rules are derived. This two-step approach determines the bases for non-redundant association rules and thus pruning redundancy, and speeding up the computation, especially when the minimum support threshold is low.

Formal Concept Analysis can also be used as an unsupervised learning technique for conceptual clustering, which generates simultaneously the descriptions of the clusters. However, a concept lattice generated from a formal context is often quite complicated, and in addition, the number of formal concepts in a concept lattice is usually quite large. This makes it difficult for discovering knowledge from a concept lattice. In [STBL01], it proposed a conceptual clustering method using the iceberg concept lattice. The main idea is to use different minimum support thresholds to automatically group formal concepts and construct the nested line diagram of iceberg concept lattices to visualize the clustering results. The derived concept hierarchies can be used for many applications such as database marketing and ontology learning.

The graphical representation of the concept lattice is extremely useful for discovering and understanding conceptual relationships between the given data for information retrieval [KC01]. It can be treated as a querying structure for a database. In a concept lattice, each concept denotes a class of equivalent queries of attributes and the concept ordering denotes query refinement. The concept lattice permits pre-calculation of results for each query, pre-calculation of refinement possibilities, and analysis of query/result behavior statistically. From this point of view, the concept lattice is a complete set of queries and results. The management system TOSCANA proposed in [VW95] has been used as a knowledge discovery tool in various research and commercial projects. This type of conceptual data systems is referred to as conceptual information systems. For a selected conceptual scale that is represented as a

Figure 5.2: The proposed approach.

collection of formal contexts, TOSCANA displays a line diagram of the corresponding concept lattice indicating all objects stored in the database with their relationships to the attributes of a selected scale, thereby allowing users to navigate through the data and analyzing specific sets of objects.

In this research, we incorporate fuzzy logic into FCA for constructing user behavior models from web usage logs.

5.4 Discovering Periodic Web Access Patterns

In this section, we propose an approach for constructing Personal Web Usage Lattice for an individual user from web usage logs and extracting the periodic web access patterns of the user from the Personal Web Usage Lattice.

Figure 5.2 shows the proposed approach, which consists of the following major steps:

- *Preprocessing.* This step preprocesses the original web usage logs and identifies all personal access sessions of individual users.
- *Constructing Personal Web Usage Lattice.* This step constructs the Personal Web Usage Lattice from the personal access sessions of a user to represent the user's periodic web access activities and hierarchical relationships among these activities.
- *Extracting Periodic Access Patterns.* This step extracts periodic access patterns from the Personal Web Usage Lattice of the user.

5.4.1 Preprocessing

In the current Web environment, web usage logs (including the web server logs, browser logs and proxy logs) record access requests from users to one or multiple websites as a sequence of requested URLs with timestamps. In this research, we focus on web server logs.

Table 5.2: An example of a semantically enriched web usage log.

UserID	Timestamp	URL	Semantic Annotation
User1	21/May/2005 08:20:01	URL1	#Topic1, #Topic2, #Topic3, ...
User1	21/May/2005 08:22:32	URL2	#Topic7, #Topic3, #Topic5, ...
User2	21/May/2005 08:22:50	URL7	#Topic1, #Topic3, #Topic8, ...
User1	21/May/2005 08:27:30	URL3	#Topic3, #Topic1, ...
User1	21/May/2005 09:10:02	URL5	#Topic7, #Topic8, #Topic3, ...
User2	21/May/2005 09:24:34	URL4	#Topic3, #Topic7, ...

The URLs recorded in web usage logs contain little semantic information about the web contents accessed by users. This makes it difficult to be used for the understanding of users' actual access behaviors, interests and intentions. To overcome this problem, we assume that each requested URL in web usage logs has been annotated with the corresponding semantic information, i.e., one or more predefined topics or categories, such as News, Sports and Entertainment. Such task can be done easily by the designer or administrator of the website manually or semi-automatically. An example of a semantically enriched web usage log is given in Table 5.2. As shown in Table 5.2, the web usage log contains several fields on user identification (UserID), request time (Timestamp), requested URLs (URL) and topics (Semantic Annotation). The user ID can be stored for access sessions of registered users. The semantic annotation indicates the predefined topics that are presented in the web page of the corresponding requested URL.

Each entry in the web usage log can simply be interpreted as "A certain user has accessed specific resources at a specific time". If the user has accessed specific resources periodically, we can consider that the user has a *web access activity* (or periodic web access pattern), which can be interpreted as "A certain user is interested in specific resources during a specific period". Therefore, it is appropriate to use a set of *periodic attributes* M_p and *resource attributes* M_r to represent web access activities.

In this research, we have defined eight real-life temporal concepts, namely *Early Morning*, *Morning*, *Noon*, *Early Afternoon*, *Late Afternoon*, *Evening*, *Night* and *Late Night*, as periodic attributes. Alternatively, we can also use days of week (e.g., Monday, Tuesday, etc.) and other real-life temporal concepts (e.g., weekdays, weekend, or months) as periodic attributes for more general purposes. In practice, each request time recorded in web usage logs is the local time of the web server. But it is also possible to convert each request time into the corresponding local time of the user's location using the client IP address to determine the time zone.

All predefined topics for a certain website can be regarded as resource attributes for describing web access activities. The selection of the appropriate domain-specific topics depends on the applications. For example, the product names in product catalogs can be used for e-commerce’s websites, whereas the categories of news can be used for news websites.

In data preprocessing, we need to process the original web usage logs in order to identify all personal access sessions for each individual user. Similar preprocessing tasks, i.e., data cleaning, user identification and session identification, for traditional web server logs discussed in [CMS99] can be used in this step.

Data cleaning. In the original web usage logs, not all records are necessary for mining purpose. We can clean out unnecessary entries according to the topics in the “Semantic Annotation” field based on the set of selected resource attributes, i.e., M_r . Entries in web usage logs are regarded as useful when the topics of their requested URLs refer to at least one resource attribute in M_r . All other records should be discarded from the web usage logs, which include picture files, scripts and other invalid documents.

Note that we assume all entries in the web usage logs only record successful requests of users. For web usage logs containing all requests from users, all entries of unsuccessful requests should also be discarded.

User identification. For analyzing personal access behavior, unique users must be identified using the “UserID” field. This means the periodic personalized web services will be provided only to registered users of the website. In addition, all requests from anonymous users can be treated as a pseudo-user “Anonym”.

Session identification. For requests from a user that span a long period of time, it is very likely that the user has performed more than one access activity. The goal of session identification is to divide web usage logs of each user into individual access sessions. The simplest approach is to set a timeout threshold. If the difference between the timestamps of two consecutive requests from the same user is greater than the timeout threshold, it could be considered that a new access session has started. Here, we use 30 minutes as the default timeout threshold.

A user access session $S = \langle (URL_1, t_1), (URL_2, t_2), \dots, (URL_n, t_n) \rangle$ is a sequence of requested URL_i with timestamp t_i ($1 \leq i \leq n$). Note that it is not necessary that $URL_i \neq URL_j$ for $i \neq j$ in S , that is repeat of requested URLs is allowed, because the same URLs may contain different contents at different request times. In general, the time spent by a

user for a URL can be used to indicate the level of interest that the user has in the content of that URL. The duration d_i of URL_i can be estimated simply as $d_i = (t_{i+1} - t_i)$. For the last requested URL URL_n in each user access session that does not have “ t_{n+1} ”, we have used the average duration of the current session for estimating its duration, i.e., $d_n = (d_1 + d_2 + \dots + d_{n-1}) / (n - 1) = (t_n - t_1) / (n - 1)$. To compute d_n , we need $n > 1$, i.e., we only retain user access sessions which contain more than one requested URL. Furthermore, we can evaluate the start and end time of S as t_1 and $(t_n + d_n)$ respectively. For subsequent processing, we ignore the date information in the session start and end time, and convert them into a value within the interval $[0, 24]$. For example, the session start time “21/May/2005 08:20:01” is converted into 8.33. We define the period of the user access session S as follows.

Definition 5.12. A *period* of a user access session S is defined as a continuous time interval with a *session start time* $ts(S) \in [0, 24]$ and a *session end time* $te(S) \in [0, 24]$, denoted as

$$p(S) = \begin{cases} [ts(S), te(S)], & \text{if } ts(S) \leq te(S) \\ [0, te(S)] \cup [ts(S), 24], & \text{otherwise} \end{cases}.$$

Suppose that each URL_i in the user access session S is associated with a set of resource attributes $M_{ri} \subseteq M_r$ for representing the semantics of the content of URL_i . As such, each user access session can be treated as a sequence of sets of resource attributes M_{ri} instead of a sequence of individual URL_i ($1 \leq i \leq n$), and is denoted as $S = \langle (M_{r1}, t_1, d_1), (M_{r2}, t_2, d_2), \dots, (M_{rn}, t_n, d_n) \rangle$. The total duration for each resource attribute $m_k \in M_r$, which can be used for estimating the level of user interest in that resource during the user access session S , can be computed as

$$d(S, m_k) = \sum_{i=1}^n \alpha_{ki} d_i, \text{ where } \alpha_{ki} = \begin{cases} 1, & \text{if } m_k \in M_{ri} \\ 0, & \text{otherwise} \end{cases} \text{ for } 1 \leq i \leq n.$$

5.4.2 Constructing Personal Web Usage Lattice

In this step, we identify a user’s web access activities and construct a Personal Web Usage Lattice from the personal access sessions of the user. Fuzzy logic is incorporated into Formal Concept Analysis for representing both periodic attributes and resource attributes.

To do this, we first construct the Web Usage Context for a user from his preprocessed user access sessions.

Definition 5.13. A fuzzy periodic *Web Usage Context* is $K = (G, M_p, M_r, I)$, where G is a set of *user access sessions* for a user, M_p is a set of *periodic attributes*, M_r is a set of *resource*

Figure 5.3: Member function $\mu_p(t, m_p)$.

attributes and $I = R(G \times (M_p \cup M_r))$ is a fuzzy set on the domain of $G \times (M_p \cup M_r)$ to represent fuzzy relations between user access sessions $g \in G$ and attributes $m \in M_p \cup M_r$. Each fuzzy relation $R(g, m) \in I$ is represented by a membership value $\mu(g, m) \in [0, 1]$, where

$$\mu(g, m) = \begin{cases} \mu_p(g, m), & \text{if } m \in M_p \\ \mu_r(g, m), & \text{if } m \in M_r \end{cases}.$$

Based on the above definition, each user access session $g \in G$ can also be denoted as a fuzzy set on the domain of $M_p \cup M_r$, i.e., $g = \{m, \mu(g, m) \mid m \in M_p \cup M_r\}$.

The membership value $\mu_p(g, m_p)$ for a periodic attribute $m_p \in M_p$ in a user access session $g \in G$ can be computed using the period of g , i.e., $p(g)$. In this research, the member function is defined as $\mu_p(g, m_p) = \max_{t \in p(g)} \{\mu_p(t, m_p)\}$, where $\mu_p(t, m_p)$ is defined in Figure 5.3, which is modified from [OS03].

The membership value $\mu_r(g, m_r)$ for a resource attribute $m_r \in M_r$ in a user access session $g \in G$ can be computed using the total duration of m_r , i.e., $d(g, m_r)$. In this research, the member function is defined as

$$\mu_r(g, m_r) = \begin{cases} 0, & \text{if } z(g, m_r) < \frac{1}{2}Z(m_r) \\ \frac{2z(g, m_r)}{Z(m_r)} - 1, & \text{if } \frac{1}{2}Z(m_r) \leq z(g, m_r) \leq Z(m_r) \\ 1, & \text{if } z(g, m_r) > Z(m_r) \end{cases},$$

where $z(g, m_r) = \frac{d(g, m_r)}{te(g) - ts(g)}$ and $Z(m_r) = \frac{\sum_{g_k \in G} d(g_k, m_r)}{\sum_{g_k \in G} (te(g_k) - ts(g_k))}$.

$Z(m_r)$ is defined as the proportion of the total duration of accessing the resource m_r in all user access sessions of the user, which indicates the user's global interest of the resource m_r . $z(g, m_r)$ is defined as the proportion of the duration of accessing the resource m_r within

Figure 5.4: Member function $\mu_r(g, m_r)$.

Table 5.3: A cross table of an example fuzzy periodic Web Usage Context of a user.

SID	P1	P2	P3	R1	R2	R3
S1	0.6	0.5	-	0.8	-	0.6
S2	-	0.4	0.8	-	-	0.9
S3	-	-	1.0	-	0.7	0.5
S4	-	0.8	0.6	0.8	0.6	0.5
S5	-	-	1.0	-	0.9	-

the user access session g , which indicates the user's local interest of the resource m_r . Figure 5.4 shows the member function $\mu_r(g, m_r)$.

A fuzzy periodic Web Usage Context can be represented by a cross table with rows indicating user access sessions and columns indicating the periodic and resource attributes. A membership value $\mu(g, m) \in [0, 1]$ in row g and column m indicates the fuzzy relation between the user access session g and attribute m .

Table 5.3 shows an example Web Usage Context of a user, which consists of five user access sessions, three periodic attributes “P1 (Late Afternoon)”, “P2 (Evening)” and “P3 (Night)”, and three resource attributes “R1 (Sports)”, “R2 (Games)” and “R3 (Chat)”. Note that “-” indicates the corresponding membership value is zero.

Based on the fuzzy periodic Web Usage Context, the Personal Web Usage Lattice can be constructed.

Definition 5.14. Given a Web Usage Context $K = (G, M_p, M_r, I)$, we define the set of attributes common to user access sessions in $A \subseteq G$ as $A^* = \{m \in M_p \cup M_r \mid \forall g \in A : \mu(g, m) > 0\}$, and the set of user access sessions which have all the same attributes in $B \subseteq M_p \cup M_r$ as $B^* = \{g \in G \mid \forall m \in B : \mu(g, m) > 0\}$.

Definition 5.15. Given a Web Usage Context $K = (G, M_p, M_r, I)$, the *fuzzy support* of a

set of attributes $B \subseteq M_p \cup M_r$ and $B \neq \emptyset$ is defined as

$$Sup(B) = \frac{\sum_{g \in B^*} (\mu_p(g) \times \mu_r(g))}{|G|},$$

where

$$\mu_p(g) = \begin{cases} \min_{m_p \in (B \cap M_p)} \{\mu_p(g, m_p)\}, & \text{if } B \cap M_p \neq \emptyset \\ 1, & \text{otherwise} \end{cases},$$

$$\mu_r(g) = \begin{cases} \min_{m_r \in (B \cap M_r)} \{\mu_r(g, m_r)\}, & \text{if } B \cap M_r \neq \emptyset \\ 1, & \text{otherwise} \end{cases}.$$

Definition 5.16. Given a Web Usage Context $K = (G, M_p, M_r, I)$, if there exists a pair (A, B) with $A \subseteq G$, $B \subseteq M_p \cup M_r$ ($B \cap M_p \neq \emptyset$ and $B \cap M_r \neq \emptyset$), $A^* = B$ and $B^* = A$, then $v(B) = \{m, \mu(B, m) \mid m \in B\}$, a fuzzy set on B , is called a *web access activity*, where $\mu(B, m) = \max_{g \in B^*} \{\mu(g, m)\}$. We also define $v(B) = v_p(B) \cup v_r(B)$, where $v_p(B) = \{m_p, \mu(B, m_p) \mid m_p \in B \cap M_p\}$ and $v_r(B) = \{m_r, \mu(B, m_r) \mid m_r \in B \cap M_r\}$ represent two fuzzy subsets of $v(B)$ on domains of $B \cap M_p$ and $B \cap M_r$ respectively. In addition, $v(\emptyset) = \emptyset$ is defined as a *virtual web access activity*.

$W_P = \{v(B_i)\}$ denotes the set of all web access activities of a user. $|W_P|$ is the total number of web access activities.

A web access activity represents a periodic web access behavior of a user, i.e., it is an implication from periodic attributes to resource attributes, which is a special kind of association rules. Similar to association rule mining, a “support” value should be given to reflect the quality of the discovered web access activity. Since we use fuzzy set theory, we define it as “fuzzy support”.

Definition 5.17. The *fuzzy support* of a web access activity $v(B)$ is defined as $Sup(v(B)) = Sup(B)$ and the *fuzzy confidence* of $v(B)$ is defined as $Conf(v(B)) = prob((B \cap M_r) \mid (B \cap M_p)) = \frac{Sup(B)}{Sup(B \cap M_p)}$, where $prob(\cdot \mid \cdot)$ is a conditional probability. For virtual web access activity $v(\emptyset)$, we define $Sup(v(\emptyset)) = 1.0$ and $Conf(v(\emptyset)) = 1.0$.

Definition 5.18. For two web access activities of a user $v(B_i)$, $v(B_j) \in W_P$, $v(B_i)$ is a *sub-activity* of $v(B_j)$, denoted as $v(B_i) <_{W_P} v(B_j)$, if and only if $B_j \subset B_i$. Equivalently, $v(B_j)$ is a *super-activity* of $v(B_i)$. $<_{W_P}$ is a partial order on W_P , called *activity relationship*. In particular, if $v(B_i) <_{W_P} v(B_j)$ and there is no $v(B_k) \in W_P$ ($B_k \neq B_i$ and $B_k \neq B_j$) such that $v(B_i) <_{W_P} v(B_k) <_{W_P} v(B_j)$, then $v(B_i)$ is a *direct sub-activity* of $v(B_j)$, and $v(B_j)$ is

Figure 5.5: Personal Web Usage Lattice.

a *direct super-activity* of $v(B_i)$. We denote this as $v(B_i) \prec_{W_P} v(B_j)$. $\prec_{W_P}$ is called a *direct activity relationship*.

Obviously, the virtual web access activity $v(\emptyset)$ is the super-activity of all other web access activities.

Definition 5.19. A *Personal Web Usage Lattice* based on a Web Usage Context $K = (G, M_p, M_r, I)$ of a user is $L_P = (W_P, \prec_{W_P})$, where W_P is the set of all web access activities, and $\prec_{W_P}$ is a partial order on W_P to represent the hierarchy of web access activities.

Figure 5.5 shows the Personal Web Usage Lattice obtained from the Web Usage Context given in Table 5.3. Each node in the figure represents a web access activity with the corresponding membership values of its attributes on the left and the fuzzy support and confidence values on the right. Each edge represents a direct activity relationship. Note that one virtual node (virtual web access activity $v(\emptyset)$) is added as the upper bound (top) of the lattice.

Some efficient approaches [STB⁺02, NR99, GMA95] have been proposed in traditional FCA for computing non-fuzzy concept lattices from the corresponding non-fuzzy formal contexts. However, they cannot process the proposed fuzzy periodic Web Usage Context and construct the Personal Web Usage Lattice directly. In this research, we have enhanced a traditional non-fuzzy concept lattice construction tool, called Galicia Lattice Builder

Table 5.4: A cross table of the formal context converted from the example Web Usage Context given in Table 5.3.

SID	P1	P2	P3	R1	R2	R3
S1	×	×		×		×
S2		×	×			×
S3			×		×	×
S4		×	×	×	×	×
S5			×		×	

(<http://www.iro.umontreal.ca/galicia/>), to compute the Personal Web Usage Lattice from the corresponding Web Usage Context. The main tasks are illustrated as follows.

At first, we need to convert the Web Usage Context into the traditional non-fuzzy formal context (defined by Definition 5.8) as the input of the Galicia Lattice Builder. Let $G' = G$ and $M' = M_p \cup M_r$. If $\mu(g, m) > 0$ ($m \in M_p \cup M_r$ and $g \in G$) in the Web Usage Context, then $(g, m) \in I'$ in the corresponding formal context $K' = (G', M', I')$. Table 5.4 shows the formal context converted from the example Web Usage Context given in Table 5.3. Then, we use the Galicia Lattice Builder to generate the corresponding non-fuzzy concept lattice.

The concept lattice of the formal context given in Table 5.4 is shown in Figure 5.6. Each node in the figure represents a formal concept with the corresponding intent (a set of periodic and resource attributes) on the left and extent (a set of user access sessions) on the right. Comparing Figure 5.5 with Figure 5.6, we can see that the Personal Web Usage Lattice is a sub-lattice of the corresponding concept lattice with fuzzy membership, fuzzy support and fuzzy confidence, which can also be verified from their definitions. We first delete formal concepts with only periodic or resource attributes or with empty extent (i.e., nodes 1, 2 and 11 in Figure 5.6). After that, all remained concepts can be mapped into web access activities. Since the extent of each formal concept contains all session IDs of the corresponding user access sessions in the original Web Usage Context, it is easy to compute the fuzzy memberships of attributes, fuzzy support and fuzzy confidence according to Definitions 5.16 and 5.17.

The algorithm **Personal_Web_Usage_Lattice_Construction** for constructing the Personal Web Usage Lattice from the corresponding Web Usage Context of a user is given in Algorithm 5.1. In practice, other efficient approaches can also be used for computing the traditional non-fuzzy concept lattice from the corresponding non-fuzzy formal context in this step instead of the Galicia Lattice Builder.

Figure 5.6: The concept lattice of the formal context given in Table 5.4.

Complexity analysis: For each web access activity (i.e., the formal concept in the concept lattice satisfied with the condition in Line 3 of Algorithm 5.1), we need one scan of all user access sessions recorded in its extent (i.e., a subset of the set of all user access sessions G) to compute its fuzzy memberships, fuzzy support and fuzzy confidence. Assume the Personal Web Usage Lattice has a total of $|W_P|$ web access activities, then computing their fuzzy memberships, fuzzy supports and fuzzy confidences need to scan at most a total of $|W_P| \times |G|$ user access sessions. Usually, $|W_P|$ is far smaller than $|G|$. Thus, the total cost for constructing the Personal Web Usage Lattice from the corresponding non-fuzzy concept lattice is $O(|G|)$.

5.4.3 Extracting Periodic Association Access Patterns

From the Personal Web Usage Lattice, inference rules can be extracted to deduce the user's periodic association access patterns. As fuzzy knowledge is stored in the Personal Web Usage Lattice, fuzzy logic [Zad75] is applied to infer association access patterns of the user. Each access pattern is represented in the form of fuzzy propositions [KY95], which are characterized by the canonical form "If x is A , then y is B ", where x and y are variables whose values

Algorithm 5.1 Personal_Web_Usage_Lattice_Construction(K)**Input:** $K = (G, M_p, M_r, I)$ - a Web Usage Context of a user**Output:** $L_P = (W_P, <_{W_P})$ - a Personal Web Usage Lattice of a user**Process:**

```

1: Initialize  $W_P \leftarrow \{\emptyset\}$  and  $<_{W_P} \leftarrow \{\emptyset\}$ 
2: Construct the formal context  $K' = (G', M', I')$ , where  $G' = G$ ,  $M' = M_p \cup M_r$ ,  $I' = \{(g, m) | \mu(g, m) > 0, \text{ where } m \in M_p \cup M_r \text{ and } g \in G\}$ 
3: Construct the concept lattice  $L_C = (C, <_C) \leftarrow \mathbf{Galicia}(K')$ 
4: for all  $(A_i, B_i) \in C$  do
5:   if  $B_i \cap M_p \neq \emptyset$  and  $B_i \cap M_r \neq \emptyset$  and  $A_i \neq \emptyset$  then
6:      $W_P \leftarrow v(B_i)$ 
7:     for all  $(A_i, B_i) <_C (A_j, B_j)$  do
8:       if  $B_j \cap M_p \neq \emptyset$  and  $B_j \cap M_r \neq \emptyset$  then
9:          $<_{W_P} \leftarrow v(B_i) <_{W_P} v(B_j)$ 
10:      else
11:         $<_{W_P} \leftarrow v(B_i) <_{W_P} v(\emptyset)$ 
12:      end if
13:    end for
14:    Compute  $Sup(v(B_i))$  and  $Conf(v(B_i))$ 
15:  end if
16: end for
17: return  $L_P$ 

```

are in sets X and Y respectively, and A and B are fuzzy sets on X and Y respectively. Such propositions are also known as fuzzy "IF-THEN" rules, and can be simply denoted as $A \Rightarrow B$. The Personal Web Usage Lattice gives *periodic association access patterns*, which can be extracted from each web access activity directly.

Definition 5.20. Given a Personal Web Usage Lattice $L_P = (W_P, <_{W_P})$, each personal web access activity $v(B) = v_p(B) \cup v_r(B) \in W_P$ (excluding virtual personal web access activities) can be represented as a *periodic association access pattern*, which is in the form of " $v_p(B) \Rightarrow v_r(B)$ ". We also define the fuzzy support as $Sup(v_p(B) \Rightarrow v_r(B)) = Sup(v(B))$ and the fuzzy confidence as $Conf(v_p(B) \Rightarrow v_r(B)) = Conf(v(B))$.

For example, for the activity $\{P2:0.8, R1:0.8, R3:0.6\}$ given in Figure 5.5, a periodic association access pattern " $\{Evening(0.8)\} \Rightarrow \{Sports(0.8), Chat(0.6)\}$ " with $Sup = 0.14$ and $Conf = 0.41$ can be extracted. It can be interpreted as "The user has interests in resources on Sports and Chat during the period of Evening". The associated fuzzy membership values can provide additional information on describing periodic association access patterns. The fuzzy support and confidence indicate the quality of such periodic association access patterns. Table 5.5 shows the eight periodic association access patterns that are extracted from the

Table 5.5: Periodic access patterns.

Periodic Association Access Patterns	Sup	Conf
$\{Evening(0.8)\} \Rightarrow \{Chat(0.9)\}$	0.21	0.62
$\{Night(1.0)\} \Rightarrow \{Chat(0.9)\}$	0.30	0.45
$\{Night(1.0)\} \Rightarrow \{Games(0.9)\}$	0.39	0.58
$\{Evening(0.8)\} \Rightarrow \{Sports(0.8), Chat(0.6)\}$	0.14	0.41
$\{Evening(0.8), Night(0.8)\} \Rightarrow \{Chat(0.9)\}$	0.13	0.66
$\{Night(1.0)\} \Rightarrow \{Games(0.7), Chat(0.5)\}$	0.16	0.24
$\{Late_Afternoon(0.6), Evening(0.5)\} \Rightarrow \{Sports(0.8), Chat(0.6)\}$	0.06	0.60
$\{Evening(0.8), Night(0.6)\} \Rightarrow \{Sports(0.8), Games(0.6), Chat(0.5)\}$	0.06	0.30

example Personal Web Usage Lattice.

Periodic association access patterns indicate the resources the user is interested in for a specific period of time such as morning and afternoon. The number of periodic association access patterns of a user may be quite complicated and huge. This is mainly due to the large number of web access activities generated with low support values. Therefore, it will be costly if all periodic association access patterns are generated and used. To overcome this problem, the periodic association access patterns should be filtered to retain only those interesting patterns, which are more important for describing the periodic web access behavior of the user.

Definition 5.21. Given a minimum support $MinSup \in [0, 1]$ and a minimum confidence $MinConf \in [0, 1]$, we call a periodic association access pattern *interesting*, if its fuzzy support value is not less than $MinSup$ and its fuzzy confidence value is not less than $MinConf$.

Given the minimum support $MinSup = 0.1$ and $MinConf = 0.15$, only six interesting periodic association access patterns are extracted.

The algorithm **Periodic.Pattern.Extraction** for extracting interesting periodic association access patterns from a Personal Web Usage Lattice of a user is given in Algorithm 5.2.

Complexity analysis: To extract periodic association access patterns from a Personal Web Usage Lattice, we need to scan all web access activities once. Thus, the cost for extracting periodic association access patterns is $O(|W_p|)$. Compared with the traditional association rule mining approaches [AS94, HPY00, AAP01], our approach should be more costly due to the construction of the Personal Web Usage Lattice. However, the Personal Web Usage Lattice is a useful and compact data structure for storing closed fuzzy association rules (i.e., periodic association access patterns). In Chapter 6, we will show how to utilize the special

Algorithm 5.2 Periodic_Pattern_Extraction(L_P , $MinSup$, $MinConf$)**Input:** $L_P = (W_P, <_{W_P})$ - a Personal Web Usage Lattice of a user $MinSup$ - a minimum support $MinConf$ - a minimum confidence**Output:** PP - a set of interesting periodic association access patterns**Process:**

- 1: Initialize $PP \leftarrow \{\emptyset\}$
- 2: **for all** $v(B) \in W_P$ ($v(B) \neq \emptyset$) **do**
- 3: **if** $Sup(v(B)) \geq MinSup$ **and** $Conf(v(B)) \geq MinConf$ **then**
- 4: $PP \leftarrow (v_p(B) \Rightarrow v_r(B))$
- 5: **end if**
- 6: **end for**
- 7: **return** PP

structure of the lattice model to reduce the search space for efficient pattern matching.

5.5 Visualizing Periodic Association Access Patterns

The Personal Web Usage Lattice contains knowledge on periodic web access behavior of individual users, i.e., periodic association access patterns. In this section, we present an approach for visualizing the periodic association access patterns.

Sometimes, it is useful for web administrators or users to view the periodic association access patterns for better understanding a particular user's web access behavior. For example, this is particularly useful for the client-side logs mining for parents understanding the web access behavior of their children [ZHF05]. However, the discovered web access patterns are presented as rules with associated fuzzy membership values for periodic and resource attributes, and fuzzy support and confidence values. These are machine-readable representations, but they are not easy for human interpretation. To help users, especially casual users, to understand the access patterns, we have mapped these values into some meaningful terms for viewing purposes.

The fuzzy set of periodic attributes of each periodic association access pattern is converted into certain time intervals using an inverse function of the membership function of periodic attributes given in Figure 5.3. For example, $\{P2:0.8\}$, $\{P1:0.6, P2:0.5\}$ and $\{P3:1.0\}$ represent the time intervals shown in Figure 5.7(a), Figure 5.7(b) and Figure 5.7(c) respectively.

Each fuzzy membership value of a resource attribute is converted into one of the interest terms "fairly interest", "interest", "very interest" and "absolutely interest" to indicate the

(a) $\{P2 : 0.8\}$

(b) $\{P1 : 0.6, P2 : 0.5\}$

(c) $\{P3 : 1.0\}$

Figure 5.7: Converting fuzzy set of periodic attributes into time intervals.

Figure 5.8: Converting fuzzy membership values of resources into meaningful interest levels.

(a) Frequency representation.

(b) Quality representation.

Figure 5.9: Converting fuzzy support and confidence into frequency and quality levels.

interest levels of the user. This is illustrated in Figure 5.8.

Fuzzy support (Sup) and fuzzy confidence ($Conf$) reflect the usefulness and certainty of the discovered web access patterns respectively. Given the minimum support threshold ($MinSup$) and the minimum confidence threshold ($MinConf$), we can derive all frequent ($Sup \geq MinSup$) and certain ($Conf \geq MinConf$) web access patterns. For fuzzy support, we convert it into one of the terms “fairly frequent”, “frequent”, “very frequent” and “absolutely frequent” as illustrated in Figure 5.9(a) to represent the “Frequency” of the web access patterns. For fuzzy confidence, we convert it into one of the terms “fairly true”, “true”, “very true” and “absolutely true” as illustrated in Figure 5.9(b) to represent the “Quality” of the web access patterns.

In this research, we have applied the proposed approach to Dataset_2, i.e., the web server logs of “North Latitude One BBS” (<http://bbs.nlonet.net>) for the period between 01-May-2005 and 31-May-2005. The web forum is mainly used by graduate students from Nanyang Technological University, Singapore. Most users access this web forum from static IP addresses. As such, individual users can be identified based on the client-side IP addresses.

The website semantics of the web forum are used to annotate the web server logs into semantic enriched web usage logs. Each requested URL can be mapped into some of the 64 topics quite directly. We have applied the proposed approach to analyze the periodic association access patterns of the top 25 frequent access users, who have more than 50 access sessions in the 31 days of May-2005.

Figure 5.10 shows the periodic association access patterns of a particular user accessing from the IP address “155.69.144.213”. A total of 67 access sessions are recorded. And 48 periodic association access patterns are extracted with $MinSup = 0.05$ and $MinConf = 0.50$. These patterns contain very useful insight on periodic-based web access behavior of that user. For example, we can learn that the user likes to access the resource *Singapore* in the *afternoon* (during 14:30:00 to 17:30:00). The discovered patterns in Figure 5.10(a) are sorted by “Quality” in descending order. In addition, we can also show all patterns by sorting the “Period” in descending order as shown in Figure 5.10(b). Through different views of a user’s access behavior, we can learn the user’s interests from morning to night on a daily basis.

5.6 Summary

In this chapter, we have proposed a fuzzy FCA based approach for discovering periodic association access patterns of individual users from web usage logs. In addition, we have also presented an approach for visualizing periodic web access behavior of an individual user. The discovered periodic association access patterns in our proposed approach are quite different from other web usage mining techniques that mainly discover simple statistical information on access behaviors of individual users or user groups. As the discovered periodic association access patterns are extracted from the Personal Web Usage Lattice model, they can also be used for providing personalized web services. For example, as periodic association access patterns can be regarded as a list of resources that is most relevant to a user’s interests for that periods of time, personalized web services for certain time periods can be provided according to the ranked list of resources based on their fuzzy membership values. In Chapter 6, we apply the Personal Web Usage Lattice model to support periodic web personalization. In addition, we also propose an approach for Web Usage Ontology generation based on the Personal Web Usage Lattice model, which will be discussed in Chapter 7.

(a) Sorted by Quality.

(b) Sorted by Period.

Figure 5.10: Visualizing periodic association access patterns.

Chapter 6

Periodic Web Personalization

As mentioned in Chapter 5, *periodic web access patterns* are very important for understanding users' access behaviors and supporting effective personalized web services. With periodic web access patterns of a user, we can easily deduce which resources that the user is most probably interested in during a specific time period without the use of the user's current access information, which is required in most existing approaches. Then, personalized resources can be prepared in advance and provided to the user during that specific period. We call such task as *Periodic Web Personalization*.

Most of the existing web personalization approaches [MDLN01, LAR02, MGPL04, MDLN02, GH03, ZHC04, Mob99] can be referred to as non-periodic approaches, which use a user's current access information (request time and requested resources) for determining which resources should be recommended next or in near future to the user. However, the user's current requested resources sometimes may not be his actual interest or target, but some intermediate pages or noise. Such inaccurate prior knowledge would significantly affect the effectiveness of personalized web services. Different from non-periodic approaches, the periodic web personalization approach avoids the use of the user's current access information, but uses a specific time period to determine personalized resources. Moreover, in non-periodic approaches, personalized resources cannot be prepared for the user in advance, since they are determined according to the user's current access information. In other words, the personalized resource preparation must be efficient and simple enough in order to provide personalized resources to the user in real-time. However, for periodic web personalization, the personalized resource preparation can be performed in advance. This makes it possible to perform more costly personalized resource preparation in personalized web services. For example, a news website can collect news from different sources and prepare personalized morning news for

each registered user during night time, and then, deliver it to the users in the next morning.

Recent related research work mainly focused on investigating techniques for mining periodic patterns, rather than applying such patterns in practical applications. Some of them gave a brief discussion on the potential applications of the periodic patterns. For example, [ORS98, RMS98] mentioned that periodic association rules can be used for trend analysis, prediction, forecasting and decision making. Li *et al.* [LNWJ01] stated in their work that the calendar-based temporal association rules can be used to represent normal network activities in different time periods of a day. Then, the Intrusion Detection System (IDS) can be used to detect attacks to the network when abnormal network activities happen. Moreover, [HC04, LD05] also discussed that the periodic patterns can be used not only for data characterization, but also be applied for future periodicity prediction. However, to the best of our knowledge, no approaches have been proposed for periodic web personalization.

In this chapter, we propose periodic web personalization, which is based on the Personal Web Usage Lattice discussed in Chapter 5. The rest of this chapter is organized as follows. In Section 6.1, we discuss the system architecture of the proposed periodic web personalization. In Section 6.2, we propose the algorithm for generating the personalized resources for given period conditions based on the Personal Web Usage Lattice of a user. The performance of the periodic web personalization approach is evaluated in Section 6.3. Finally, a summary of this chapter is given in Section 6.4.

6.1 System Architecture

Figure 6.1 shows the architecture of the proposed periodic web personalization system. All web access requests of a user are first recorded by the Web server of the website and stored into its web usage logs. The *User Behavior Model Construction* component then constructs the Personal Web Usage Lattice model of the user using the approach proposed in Chapter 5. When the Web server gives period conditions, which may be specified by the user or the Web server, the *Personalized Resources Generation* component will generate the personalized resources from the Personal Web Usage Lattice. The Web server will then prepare and organize the corresponding resources and provide them to the user during the given time periods.

Figure 6.1: The architecture of the proposed periodic web personalization system.

6.2 Personalized Resources Generation

In practice, web users usually expect that their registered websites are able to provide personalized resources to them during certain periods according to their periodic habits and interests. For example, a user may want his favorite news website to deliver sports news to him during a certain period on every morning, but entertainment news during another period on every evening. To achieve this, the web personalization system must be able to deduce which resources the user is most probably interested in during the given periods.

The *Personalized Resources Generation* component searches the Personal Web Usage Lattice of a user for generating personalized resources according to a given period condition.

A period condition can be a time point or a time interval. The definition of a *period condition* is given as follows.

Definition 6.1. A *period condition* p_c is defined as a continuous time interval with a *start time* $ts \in [0, 24]$ and an *end time* $te \in [0, 24]$, which is denoted as

$$p_c = \begin{cases} [ts, te], & \text{if } ts \leq te \\ [0, te] \cup [ts, 24], & \text{otherwise} \end{cases}.$$

Note that a time point $t \in [0, 24]$ can be regarded as a special period condition $p_c = [t, t]$. In addition, a period condition can be converted into the corresponding *fuzzy period condition*, which is defined as follows.

Definition 6.2. The *fuzzy period condition* based on a period condition p_c is defined as a fuzzy set $P_f(p_c) = \{m_p, \mu_p(p_c, m_p) \mid m_p \in M_p \text{ and } \mu_p(p_c, m_p) = \max_{t \in p_c} \{\mu_p(t, m_p)\}\}$, where $\mu_p(t, m_p)$ was defined in Figure 5.3.

Given a period condition p_c , the ordered set of personalized resources can be generated for a user based on the corresponding Personal Web Usage Lattice. The period condition p_c can be determined by either the user or website. For example, a user can request a news website to provide personalized morning news to him during $p_c = [7.5, 8.5]$ (i.e., from 7:30 a.m. to 8:30 a.m.) every morning. The news website can set $p_c = [19.0, 22.0]$ as the period condition for providing personalized night news to all registered users. In addition, online periodic web personalization can also be achieved using the user's current request time as p_c .

In order to generate personalized resources, we first need to search the Personal Web Usage Lattice L_P for all *period-supported activities* of p_c , which are web access activities with similar periodic attributes to the fuzzy period condition $P_f(p_c)$.

Definition 6.3. The *period similarity* between a web access activity $v(B) \in W_P$ and a given period condition p_c is defined as

$$Sim_p(v(B), p_c) = \frac{|v_p(B) \cap P_f(p_c)|}{|v_p(B) \cup P_f(p_c)|}.$$

For example, suppose that the period condition $p_c = [17:00:00, 17:30:00] = [17.00, 17.50]$ and the web access activity $v(B) = \{P1:0.6, P2:0.5, R1:0.8, R3:0.6\}$, then $P_f(p_c) = \{P1:0.33, P2:0.5\}$ and $Sim_p(v(B), p_c) = \frac{|\{P1:0.6, P2:0.5\} \cap \{P1:0.33, P2:0.5\}|}{|\{P1:0.6, P2:0.5\} \cup \{P1:0.33, P2:0.5\}|} = \frac{|\{P1:0.33, P2:0.5\}|}{|\{P1:0.6, P2:0.5\}|} = \frac{0.33+0.5}{0.6+0.5} = \frac{0.83}{1.1} = 0.75$.

Definition 6.4. The *period-supported activities* of a given period condition p_c based on a Personal Web Usage Lattice $L_P = (W_P, <_{W_P})$ are defined as a set of web access activities $SA_p(p_c, L_P) = \{v(B_i) \mid v(B_i) \in W_P, v_p(B_i) \subseteq P_f(p_c) \text{ or } v_p(B_i) \supseteq P_f(p_c)\}$. For two period-supported activities $v(B_i), v(B_j) \in SA_p(p_c, L_P)$, we call that $v(B_i)$ has a higher priority than that of $v(B_j)$, denoted as $v(B_j) <_{pri} v(B_i)$, if

- $Sim_p(v(B_i), p_c) > Sim_p(v(B_j), p_c)$, or
- $Sim_p(v(B_i), p_c) = Sim_p(v(B_j), p_c)$, but $Conf(v(B_i)) > Conf(v(B_j))$, or
- $Conf(v(B_i)) = Conf(v(B_j))$, but $Sup(v(B_i)) > Sup(v(B_j))$.

For example, the web access activity $v(B) = \{P1:0.6, P2:0.5, R1:0.8, R3:0.6\}$ is one period-supported activity of the period condition $p_c = [17:00:00, 17:30:00]$ based on the

sample Personal Web Usage Lattice, because $v_p(B) = \{P1:0.6, P2:0.5\} \supseteq P_f(p_c) = \{P1:0.33, P2:0.5\}$.

From the discovered period-supported activities, we can extract personalized resources, which are defined as follows.

Definition 6.5. The *personalized resources* of a given period condition p_c based on a Personal Web Usage Lattice $L_P = (W_P, <_{W_P})$ is defined as a set of resource attributes occurring in at least one activity in $SA_p(p_c, L_P)$, i.e., $PR(p_c, L_P) = \{m_r \mid \exists v(B_i) \in SA_p(p_c, L_P), \text{ such that } m_r \in B_i \cap M_r\}$.

For example, R1 and R3 are two personalized resources of the period condition $p_c = [17:00:00, 17:30:00]$ based on the sample Personal Web Usage Lattice, because they belong to one period-supported activity $v(B) = \{P1:0.6, P2:0.5, R1:0.8, R3:0.6\}$.

In order to distinguish the importance of personalized resources to the user, we need to rank the resources according to the order of their priorities as follows.

Definition 6.6. The *period-supported activity* of each $m_r \in PR(p_c, L_P)$, which is $v(B_i) \in SA_p(p_c, L_P)$ with $m_r \in B_i \cap M_r$, is given the highest priority. For two personalized resources m_{ri} and m_{rj} , we call that m_{ri} has a higher priority than that of m_{rj} , denoted as $m_{rj} <_{pri} m_{ri}$, if

- the period-supported activity of m_{ri} has a higher priority than that of m_{rj} , or
- m_{ri} and m_{rj} have the same period-supported activity $v(B)$, but m_{ri} has a higher membership value than that of m_{rj} in $v(B)$, i.e., $\mu(B, m_{ri}) > \mu(B, m_{rj})$.

For example, for the period condition $p_c = [17:00:00, 17:30:00]$, two personalized resources $R3 <_{pri} R1$, because they belong to the same period-support activity, but R1 has a higher membership value than R3.

The algorithm **PR_Gen** for generating an ordered set of personalized resources for a given period condition p_c based on a Personal Web Usage Lattice of a user L_P is given in Algorithm 6.1. In Line 5, the algorithm **PSA_Search**, which is given in Algorithm 6.2, is called for searching all period-supported activities for p_c based on L_P . In Line 6, all period-supported activities are sorted in descending order of priority. In Line 7, since all personalized resources belong to the same period-supported activity, they are only sorted in descending order of fuzzy membership value.

Algorithm 6.2 gives the algorithm **PSA_Search** for searching period-supported activities for a given period condition p_c based on a Personal Web Usage lattice of a certain user L_P .

Algorithm 6.1 PR_Gen(p_c, L_P)**Input:** p_c - a period condition $L_P = (W_P, <_{W_P})$ - a Personal Web Usage Lattice of a user**Output:** $PR_o(p_c, L_P)$ - an ordered set of personalized resources for p_c based on L_P **Process:**

```

1: Initialize  $PR_o(p_c, L_P) \leftarrow \{\emptyset\}$ 
2: for all  $v(B) \in W_P$  do
3:    $v(B).mark \leftarrow unvisited$ 
4: end for
5: Generate period-supported activities  $SA_p(p_c, L_P) \leftarrow \mathbf{PSA\_Search}(p_c, L_P, v(\emptyset))$ 
6: for all  $v(B) \in SA_p(p_c, L_P)$  in priority descending order do
7:   for all  $m_r \in B \cap M_r$  in priority descending order do
8:     if  $m_r \notin PR_o(p_c, L_P)$  then
9:       Append  $m_r$  into  $PR_o(p_c, L_P)$ 
10:    end if
11:  end for
12: end for
13: return  $PR_o(p_c, L_P)$ 

```

To enhance the efficiency of the period-supported activity searching process, the algorithm starts searching from the top node of the lattice and performs recursive searching for only the direct sub-activities satisfying the period condition p_c .

Note that we do not need to extract periodic association access patterns from the Personal Web Usage Lattice, and then, generate personalized resources. Algorithm 6.1 and 6.2 utilize the special structure of the lattice model to reduce the search space in order to generate personalized resources efficiently.

Example: Let's consider the Personal Web Usage Lattice L_P given in Figure 5.5. Suppose the period condition $p_c = [21:30:00, 22:00:00] = [21.50, 22.00]$. Then, $P_f(p_c) = \{P3:1.0\}$. The personalized resources for p_c based on L_P is generated as follows. The searching for period-supported activities starts from the top node $v(\emptyset)$ in L_P , i.e., Activity 0 in Figure 6.2(a). We first check its first direct sub-activity Activity 1. Since Activity 1 does not satisfy the condition in Line 3 in Algorithm 6.2, the searching for this direction is terminated. Next, Activity 2 is visited and identified as a period-supported activity of p_c . Then, we scan its direct sub-activities. Activity 5 does not satisfy the condition in Line 4, so it is not a period-supported activity of p_c , but its sub-activities are required to be traversed. Accordingly, Activities 8, 6 and 3 are visited, and only Activities 6 and 3 are identified as period-supported activities. In Figure 6.2(a), the bold lines with arrows represent the above

Algorithm 6.2 $\text{PSA_Search}(p_c, L_P, v(B))$ **Input:** p_c - a period condition $L_P = (W_P, <_{W_P})$ - a Personal Web Usage Lattice of a user $v(B)$ - the current web access activity**Output:** $SA_p(p_c, L_P)$ - a set of period-support activities for p_c based on L_P **Process:**

```

1: Initialize  $SA_p(p_c, L_P) \leftarrow \{\emptyset\}$ 
2: for all direct sub-activities of  $v(B)$ , i.e.,  $v(B_i) \in W_P$  with  $v(B_i) <_{W_P} v(B)$  do
3:   if  $v(B_i).mark = \text{unvisited}$  and  $v(B_i) \cap P_f(p) \neq \emptyset$  then
4:     if  $v_p(B_i) \subseteq P_f(p_c)$  or  $v_p(B_i) \supseteq P_f(p_c)$  then
5:        $SA_p(p_c, L_P) \leftarrow \{v(B_i)\} \cup \text{PSA\_Search}(p_c, L_P, v(B_i))$ 
6:     else
7:        $SA_p(p_c, L_P) \leftarrow \text{PSA\_Search}(p_c, L_P, v(B_i))$ 
8:     end if
9:      $v(B_i).mark \leftarrow \text{visited}$ 
10:  end if
11: end for
12: return  $SA_p(p_c, L_P)$ 

```

searching process, and the activities with bold circles are period-supported activities for p_c based on L_P . We sort the period-supported activities in descending order of priority, and extract the ordered personalized resources as $PR_o(p_c, L_P) = \{R2, R3\}$, where $R3 <_{pri} R2$. Finally, we obtain the personalized resources for p_c based on L_P as “ $\{Games, Chat\}$ ”, which indicates that the user is most probably interested in resources on Games and Chat during the period of [21:30:00, 22:00:00]. According to the generated personalized resources, the corresponding resources can be prepared in advance and deliver them to the user between 21:30:00 and 22:00:00 everyday. For another example, we attempt to generate personalized resources for the period [17:00:00, 17:30:00], i.e., $p_c = [17.00, 17.50]$ and $P_f(p_c) = \{P1:0.33, P2:0.5\}$. Figure 6.2(b) shows the searching process for the period-supported activities. As a result, only Activity 7 is identified as the period-supported activity, and the generated personalized resources are “ $\{Sports, Chat\}$ ”.

Complexity analysis: To generate personalized resources, at first, we need to scan all web access activities in the Personal Web Usage Lattice (Algorithm 6.1, Lines 2 - 4) to initialize the visiting flag of each web access activity as “*unvisited*”. And then, we visit a part of web access activities (Algorithm 6.2) to find all period-supported activities. At last, we scan all returned period-supported activities (Algorithm 6.1, Line 6 - 12) to generate the final personalized resources. In the worst case, we need to scan all web access activities thrice,

Figure 6.2: The searching process for period-supported activities.

i.e., all web access activities are period-supported activities. Here, we assume equal cost for operations on each web access activity in the above three steps. Thus, the largest total cost of generating personalized resources based on the Personal Web Usage Lattice with a total of n web access activities is $O(n)$. In terms of storage space, all period-supported activities are required to be stored in the memory. Therefore, we need to keep all web access activities in the memory in the worst case.

Sometimes there are too many personalized resources generated. In our approach, only the top n personalized resources are considered for periodic web personalization. We set $n = 4$ as the default. The reason will be given in the next section on performance evaluation.

6.3 Performance Evaluation

In this section, we evaluate the performance of the proposed periodic web personalization approach.

6.3.1 Evaluation Measures

Definition 6.7. Suppose we generate an ordered set of personalized resources “ $PR_o(p_c, L_P)$ ” for a given period condition p_c based on a Personal Web Usage Lattice L_P of a user. If $PR_o(p_c, L_P) \neq \emptyset$, we call it *applicable*. Let $S = \{m, \mu(S, m) \mid m \in M_p \cup M_r\}$ be a user access session in the period $p(S)$ of that user. If $p_c \cap p(S) \neq \emptyset$, we call S a *period-supported session* of the generated personalized resources. If $\exists m_r \in PR_o(p_c, L_P)$ with $\mu_r(S, m_r) > 0$, we call S a *resource-supported session* of the generated personalized resources.

In order to objectively evaluate the feasibility of our proposed approach, we define the following two evaluation measures.

Definition 6.8. Let $PR_{all} = \{PR_1, PR_2, \dots, PR_n\}$ be a collection of sets of personalized resources for the overall web personalization, and PR_a be the subset of PR_{all} comprising all applicable sets of personalized resources. The *applicability* of the overall web personalization is defined as

$$applicability = \frac{|PR_a|}{|PR_{all}|}.$$

As the Personal Web Usage Lattice only stores typical web access activities (supported by some user access sessions in the training web usage logs), if the period-supported activities cannot be found, the generated set of personalized resources will be empty. Therefore, the

applicability measure gives a rough idea of how often applicable sets of personalized resources will be generated.

Definition 6.9. Let S_T be the set of all user access sessions for testing. $SS_p(PR_i)$ is the subset of S_T comprising all period-supported sessions of the set of personalized resources PR_i , and $SS_r(PR_i)$ is the subset of $SS_p(PR_i)$ comprising all resource-supported sessions in $SS_p(PR_i)$. The *satisfaction* of the set of personalized resources PR_i is defined as

$$satisfaction(PR_i) = \begin{cases} 0, & \text{if } SS_p(PR_i) = \emptyset \\ \frac{|SS_r(PR_i)|}{|SS_p(PR_i)|}, & \text{otherwise} \end{cases}.$$

The *satisfaction* for the overall web personalization is defined as

$$satisfaction = \frac{\sum_{PR_i \in PR_a} satisfaction(PR_i)}{|PR_a|}.$$

The satisfaction measures how likely a user is interested in one of the personalized resources in the period-supported sessions.

6.3.2 Experiments

The proposed periodic web personalization system is implemented in C++. All experiments are performed on a 3.4GHz Intel Pentium 4 PC machine with 1.0GB memory, running on Microsoft Windows XP Professional. The Dataset_2, i.e., the web usage logs collected from “North Latitude One BBS”, is used for evaluating the proposed periodic web personalization approach. The top 4 users with most user access sessions are selected as test users. We use user access sessions of the 4 users from 01-May-2005 to 20-May-2005 as the training dataset to construct the Personal Web Usage Lattices for the test users, and use user access sessions of the 4 users from 21-May-2005 to 31-May-2005 as the testing dataset to evaluate the performance of the proposed periodic web personalization approach. The session information of the training and testing datasets of the test users are listed in Table 6.1.

We have conducted two experiments to evaluate the performance of the proposed approach. In both experiments, we construct the Personal Web Usage Lattices for the four test users. As a result, the number of generated web access activities of users u13, u36, u48 and u82 are 912, 1196, 258 and 1005 respectively. We then perform the two experiments based on the Personal Web Usage Lattices.

In the first experiment, we generate personalized resources for a set of predefined period

Table 6.1: Experimental datasets.

UserID	Training Dataset (Number of Sessions)	Testing Dataset (Number of Sessions)
u13	94	38
u36	112	47
u48	94	37
u82	144	49

conditions and measure the performance based on the applicability and satisfaction measures with respect to the different durations of period conditions (from 0.5 to 4 hours) and the different numbers of personalized resources (from 1 to 6). Suppose that $[t_s, t_e] = [0, 24]$ is the whole period for periodic web personalization and $d \in (0, 24]$ is the duration of each period condition. Then, we have a total of $n = \lfloor \frac{t_e - t_s - d}{d} + 1 \rfloor$ period conditions, where each period condition $p_i = [(t_s + (i - 1) \times d), (t_s + i \times d)]$ for $1 \leq i \leq n$. For example, if $d = 4$, then we have 6 period conditions, which are $p_1 = [0, 4]$, $p_2 = [4, 8]$, $p_3 = [8, 12]$, $p_4 = [12, 16]$, $p_5 = [16, 20]$ and $p_6 = [20, 24]$. In this experiment, we set d to four different values, i.e., $d = 0.5, 1.0, 2.0$, and 4.0 .

Figure 6.3 shows the applicability of the overall web personalization based on the different durations of period conditions (from 0.5 to 4.0 hours). The experimental results have shown that the proposed periodic web personalization approach has obtained an acceptable applicability for the predefined period conditions with appropriate durations. We have achieved at least 85.4% for the user u48 with $d = 0.5$ and an average of 92.3% for all the four test users with period conditions longer than or equal to 0.5 hours.

Figure 6.4 shows the satisfaction of the overall web personalization for the different durations of period conditions (from 0.5 to 4.0 hours) based on different numbers of personalized resources (from 1 to 6). As shown in Figure 6.4, when the number of personalized resources increases, the satisfaction also increases. But, the increase is not significant after the number of personalized resources is more than 4. Although the satisfaction can be further improved with more personalized resources (e.g., 6), too many personalized resources will affect the preparation process of useful resources for users. As such, we use 4 as the default number of personalized resources for providing the periodic-based web personalized service for users of this web forum site. With at most 4 personalized resources, we have achieved the lowest satisfaction of 77.9% for the user u82 with $d = 1.0$ and the average satisfaction of 88.9%

Figure 6.3: Applicability of the overall web personalization based on the different durations of period conditions.

for all the four test users. The proposed approach has achieved very effective web personalization for the predefined period conditions. This makes it possible to perform more costly personalized resource preparation for personalized web services.

The second experiment evaluates the performance of the proposed periodic web personalization to that of the traditional non-periodic approaches for real-time web personalization, although real-time web personalized services are not the main purpose of the proposed approach. In this experiment, we compare the performance of the proposed periodic approach with the non-periodic approach proposed in [MDLN01], which is based on association rules discovered from web usage logs. We assume that real-time periodic and non-periodic web personalization approaches are performed. For the proposed periodic web personalization approach, the time of each request in the testing dataset is regarded as the period condition and used for generating periodic personalized resources, while the non-periodic approach uses the requested resource of each request in the testing dataset as the prior knowledge to generate non-periodic personalized resources by matching the requested resource against the discovered association rules.

The experimental results are given in Figure 6.5 and Figure 6.6. Figure 6.5 shows the comparison of the applicability between the proposed periodic approach and the non-periodic approach for the four test users. The experimental results have shown that the applicabilities of all users for both periodic and non-periodic approaches have achieved 100%, except for the user u48 with *applicability* = 88.2% using the periodic approach.

Figure 6.4: Satisfaction of the overall web personalization for predefined period conditions based on the number of personalized resources.

Figure 6.6 shows the comparison of the satisfaction between the proposed periodic approach and the non-periodic approach based on the different numbers of personalized resources for the four test users. As shown in Figure 6.6, the satisfaction of the proposed periodic approach is comparable with the association rule based non-periodic approach for all the four test users. As a result, we have achieved an average satisfaction of 86.5% when using 4 personalized resources for real-time periodic web personalization, and 88.3% for the real-time association rule based non-periodic approach.

The results of the above two experiments have shown that the proposed periodic approach has achieved very effective web personalization for resource preparation purpose based on both predefined and real-time period conditions.

Figure 6.5: Performance comparison based on the applicability measure.

6.4 Summary

In this chapter, we have proposed a novel web usage mining approach for supporting an effective periodic web personalization using periodic web access patterns of individual users. Different from non-periodic approaches, the proposed approach can efficiently determine which resources a user is most probably interested in during a given period based on the Personal Web Usage Lattice without the use of the user's current access information. This makes it possible to perform more costly personalized resource preparation in advance rather than in real-time. In addition, some unusual requests that have no relation with the user's intended access may affect the effectiveness of non-periodic approaches, but not that of the proposed periodic approach.

The experimental results have shown that the proposed approach has achieved very effective web personalization evaluated by the applicability and satisfaction measures for the predefined period conditions. In addition, a comparable performance has been obtained in comparison with association rule based non-periodic approach.

Apart from supporting periodic web personalization, we have also used the proposed Personal Web Usage Lattice to build Web Usage Ontology to support Semantic Web personalization. This will be discussed in Chapter 7.

(a) u13

(b) u36

(c) u48

(d) u82

Figure 6.6: Performance comparison based on the satisfaction measure.

Chapter 7

Periodic-based Semantic Web Personalization

As mentioned in Chapter 5, the URLs recorded in traditional web usage logs contain little semantic information about the web contents accessed by users. This makes it difficult to be used for the understanding of users' actual access behaviors, interests and intentions. The Semantic Web [BLHL01] provides a common framework that allows data to be shared and reused across application, enterprise and community boundaries. Ontology, which is an important component of the Semantic Web, supports the description of the semantics of web contents. Semantic web usage logs can be constructed from web usage logs in which each requested URL is annotated with semantic information from the ontology of the website.

In this chapter, we incorporate the proposed Personal Web Usage Lattice model and periodic web personalization approach into the Semantic Web environment for generating Web Usage Ontology of individual users and providing periodic-based personalized services to users on the Semantic Web.

The proposed approach is very different from other semantic web usage mining approaches [OBHG03, EVV03, MLME04, FMM03]. Unlike most current approaches that only mine semantic web usage logs for usage statistics, the proposed approach discovers periodic web access behavior of users directly from semantic web usage logs, and automatically generates web usage ontologies that are described in OWL (Web Ontology Language) [MH04] for the storing and sharing of the discovered web access activities on the Semantic Web. The Global Web Usage Ontology defines all possible activity classes of users of a semantic website, which are represented by a set of attribute properties, taxonomy properties and quality properties. An example of an activity class is given in Figure 7.5. And one Personal Web Usage Ontology

contains all actual activity instances of a user, which are derived from the user's web access activities. In other words, the Personal Web Usage Ontology is the OWL representation of the user's Personal Web Usage Lattice proposed in Chapter 5. An example of an activity instance is given in Figure 7.6. In addition, OWL-S (Web Ontology Language for Services) [MBH⁺04] is used to describe the periodic-based personalized service on the Semantic Web based on the proposed periodic web personalization approach. Given a user ID, start and end time of a period condition, an ordered set of personalized resources can be computed from the sharable Personal Web Usage Ontology of the user effectively and efficiently through the proposed Personalized Resources Generation Service. Such knowledge is very useful for supporting periodic web personalization on the Semantic Web.

The rest of this chapter is organized as follows. In Sections 7.1, 7.2, 7.3 and 7.4, we review the Semantic Web technology, semantic web usage mining techniques, ontology generation and semantic web services, respectively. Section 7.5 presents the overview of the proposed system. Section 7.6 proposed the approach for automatic generation of Personal Web Usage Ontology. Section 7.7 presents the implementation of the personalized resource generation service. Section 7.8 discusses the service specification for supporting periodic-based personalized service on the Semantic Web. Section 7.9 presents the prototype system. Finally, a summary of this chapter is given in Section 7.10.

7.1 The Semantic Web

The Semantic Web has been proposed as an extension of the current Web. The key idea is the use of machine-processable web information. Berners-Lee *et al.* [BLHL01] suggested a layered structure for the Semantic Web, which presents the development of the Semantic Web in steps, with each step building a layer on top of another. This structure follows the understanding that each step alone will provide added value, so that the Semantic Web can be realized in an incremental manner. The structure of the Semantic Web consists of the following layers:

- *Unicode/URI*. The Semantic Web uses Unicode to encode the content. Uniform Resource Identifier (URI) is a fundamental component of the current Web and is also a foundation of the Semantic Web. URI provides the ability for uniquely identifying resources as well as relationships among resources.
- *XML/Name Spaces/XML Schema*. The Extensible Markup Language (XML) is also a

fundamental component for supporting the Semantic Web. XML provides an interoperable syntactical foundation upon which the more important issue on the presentation of relationships and meanings can be built.

- *RDF/RDF Schema.* The Resource Description Framework (RDF) [KC04] family of standards leverages URI and XML to provide a stepwise set of functionality to represent the relationships and meanings.
- *Ontology Vocabulary.* An ontology [BLHL01] is a specification of a conceptualization, which is an abstract, simplified view of the world that we wish to represent for some purposes. For the Semantic Web, an ontology is a conceptualization of a domain into a human understandable, but machine readable format consisting of entities, attributes, relationships, and axioms.
- *Logic.* The logic layer is used to enhance the ontology language further and allow the specification of application-specific declarative knowledge.
- *Proof.* The proof layer involves the actual deductive process as well as the representation of proofs in markup languages from lower layers and proof validation.
- *Trust.* The trust layer is achieved by making use of digital signatures and other kinds of knowledge based on the recommendations by trusted agents, rating and certification agencies, and consumer bodies.

Despite the advantages that the Semantic Web promises, several issues have to be resolved in order to realize the full benefits of the Semantic Web. Some of the issues that are related to this research are listed below:

- *There is currently little Semantic Web content available.* Existing web content should be transferred into Semantic Web content including static HTML pages, dynamic content, multimedia, web usage data and web services.
- *Ontologies will become a key knowledge, as they allow explicating the semantics of Semantic Web content.* The commonly used ontologies for the Semantic Web must be created on the provision of adequate infrastructure for ontology development, change management and mapping.
- *Significant effort must be made to organize Semantic Web content, store it and provide the necessary mechanisms to find it.* All these tasks must be performed and coordinated in a scalable manner, as these solutions should be prepared for the huge growth of the Semantic Web.

In this chapter, we address these issues in order to support Semantic Web personalization.

7.2 Web Usage Mining for the Semantic Web

The effort behind the Semantic Web is to allow the addition of semantic annotations to web documents to enable the access and sharing of semantic knowledge. As traditional web usage logs only record requested URLs, but not the meaningful contents requested by users, it is difficult to use such logs for understanding users' actual web access behaviors and interests by both human and machine. Semantic web usage mining [SBH02] focuses on mining web usage logs for and on the Semantic Web.

In [DM02], Dai *et al.* proposed a general framework to automatically discover the domain-level aggregate profile using web usage mining techniques with domain ontology. In this approach, a website is regarded as a collocation of instances belonging to certain classes of the underlying domain ontology. Given the session-based traditional web usage data, clusters of similar user sessions are obtained based on web pages commonly accessed within those sessions using a clustering technique. Each cluster, also known as a usage profile, is transformed into a set of weighted instances, which are accessed together frequently by a group of users, according to the underlying domain ontology. The domain-level aggregate profile is then created by using the usage profiles to characterize the common interests of the users at the domain level. The aggregate profile is used for web personalization. However, the proposed framework can only characterize usage profiles for a group of users, but not on individual users. This approach of using semantics to enhance web usage mining techniques for mining traditional web usage logs has the drawback on which the mapping from the requested URLs into the ontological entities is not always reliable. As the mapping is addressed only during the log analysis process, rather than the log recording process, it is difficult to ensure the accuracy of the mappings from the requested URLs to the ontological entities. This is especially true for websites with many dynamic web pages or websites that are updated frequently. To tackle this, we can enable the web server to record the corresponding ontological entities directly into the log entries during the log recording process.

Oberle *et al.* [OBHG03] proposed a framework for semantic enrichment of web usage logs. The main idea is to map each requested URL into one or more concepts from the ontology of the underlying website. The users' actions are described at different levels of abstractions using the taxonomy of the ontology. The proposed framework applies clustering to cluster groups of sessions with specific user interests from the semantic-enhanced web usage logs. It also applies association rule mining to the semantic-enhanced web usage logs for discovering meaningful association rules. The discovered knowledge can be used to improve

the navigational structure of the website. In addition, the proposed semantic logs can also be applied to provide services of recommendation and personalization on the Semantic Web.

In [EVV03], Eirinaki *et al.* proposed the concept-logs (C-logs), which are an extended form of the web server logs by enriching each record with keywords from a taxonomy representing the semantics of the requested URLs. The C-logs are used for enhancing the web personalization process by providing a broader and more semantically focused set of recommendations. The C-logs were also analyzed in [MLME04] with MINE RULE (a query language for association rule mining) for discovering various access patterns. The discovered information includes simple statistical information on users who have visited the same web pages, most frequent crawling paths, high traffic users and anomalies. However, this approach requires users to provide queries using MINE RULE for finding the access patterns.

Fraternali *et al.* [FMM03] proposed another form of usage logs called conceptual logs which are standard web usage logs enriched with semantic elements. The conceptual logs are created by combining the server log data with the conceptual schema of the web application. XML is used as the format for representing the conceptual logs. The conceptual logs are used for quality analysis and usage evaluation of websites. The useful reports about the data access, hypertext access and navigation paths can be extracted from the conceptual logs. These reports highlight the disadvantages of the web design and can be used as suggestions for improving the design of the website.

Most of the recent approaches focus only on discovering simple usage statistics and common access patterns of user groups. Further, as the discovered knowledge is not presented in the form of ontology, it cannot be shared by other services on the Semantic Web. In this research, we aim to build Web Usage Ontology from the semantic web usage data to support personalized services on the Semantic Web.

7.3 Ontology Generation

The success of the Semantic Web greatly depends on the ontologies that structure web data for comprehensive and transportable machine understanding. Thus, ontology generation is one of the most important tasks for the realization of the Semantic Web.

Typically, an ontology consists of a finite list of terms and the relationships between these terms. The terms denote important concepts (classes of objects) of the domain, while the relationships include hierarchies of classes. An ontology may also include other information, such as properties, value restrictions, disjointness statements and specifications of

Figure 7.1: Architecture of Web Services.

logical relationships between objects. Ontology languages are semantic markup languages for defining ontologies. In this research, we use OWL (Web Ontology Language) [MH04], which was proposed as W3C Recommendation. OWL facilitates greater machine interpretability of web content that are supported by XML, RDF and RDF Schema by providing additional vocabularies along with a formal semantics.

Ontologies can be constructed manually using an ontology editor, such as Protégé [NM01] and OntoEdit [SEA⁺02]. However, manual ontology construction is a very expensive and cumbersome task, even for experts. The integration of knowledge acquisition with machine learning techniques makes it possible for generating ontology automatically or semi-automatically. Currently, many novel approaches have been investigated for generating ontology [MS01]. These include Natural Language Processing (NLP) techniques [TBGR00], association rule mining [MS00a], hierarchical clustering [CCH01] and Formal Concept Analysis (FCA) [CST03, QHFC04]. However, these works have focused mainly on constructing concept hierarchies from text documents. In particular, FCA is one of the most effective techniques for automatic ontology generation from text documents. The lattice-based concept hierarchy generated in FCA is more informative than those generated in other techniques.

To the best of our knowledge, no approaches have been proposed for automatic generation of ontology from web usage logs.

7.4 Web Services and Semantic Web Services

The Web has become a communication medium for various network applications. A *web service* [BHM⁺04] is a software system designed to support interoperable machine-to-machine interaction over a network. Web services provide a standard means of interoperability between different software applications running on a variety of platforms and/or frameworks.

The architecture of Web Services is shown in Figure 7.1, which describes three roles: Service Provider, Service Requester and Service Broker, with three basic operations: *Publish*, *Find* and *Bind*. Service Provider *publishes* the service to a Service Broker; Service Requester *finds* the required service through a Service Broker and *binds* to the service with the corresponding Service Provider.

To publish a web service, a Service Provider must create and deploy its own services in a service description language (such as the Web Services Description Language (WSDL) [CCMW01]), and register it into a service registry (such as a Universal Description Discovery and Integration (UDDI) [OAS04] Business Registry) of a Service Broker. The Service Broker indexes all registered services and provides the brokering services. A Service Requester uses the brokering services to discover its required services through the Service Broker, and then, invokes the remote services from the corresponding Service Provider by a communication protocol (such as the Simple Object Access Protocol (SOAP) [Mit03]).

At present, the use of web services requires human involvement. The Semantic Web services aim to make it automatic for the discovery, invocation, composition and monitoring of web services by providing machine-interpretable descriptions of services.

To make use of a web service on the Semantic Web, a software agent needs a computer-interpretable description of the service and the means to access it. OWL-S (OWL Web Ontology Language for Services) provides the capability to form the required ontology for web services. OWL-S uses OWL to define a set of classes and properties specific to the description of services. Building upon the SOAP and WSDL technologies, the OWL-S ontology-based web services can be dynamically invoked by other services on the Web.

In OWL-S, the ontology of services provides three essential types of knowledge about a service: *Service Profile*, *Service Model* and *Service Grounding*.

- *Service Profile*. It provides a general description about a web service that is shared to facilitate service discovery. An agent can use it to determine whether the service meets its needs, and whether it satisfies the constraints such as security, locality and quality requirements. A Service Profile can include both functional properties (inputs, outputs, preconditions, and results) and nonfunctional properties (service name, text description, contact information, service category, and additional service parameters).
- *Service Model*. It enables an agent to perform an in-depth analysis on whether the service meets its needs. The Service Model is based on the concept of *process*, which describes a service in details in terms of inputs, outputs, preconditions, results and its composition of component sub-processes.

Figure 7.2: Semantic Web personalized services.

- *Service Grounding*. It specifies the details of how to access the service such as protocol and message formats, serialization, transportation and addressing. OWL-S provides the WSDL grounding, which allows any traditional web services to be marked up as a Semantic Web service.

Therefore, to create an OWL-S based ontology for a web service, it requires the creation of the Service Model, Service Grounding and Service Profile. In this research, we use OWL-S to describe the Semantic Web service for supporting periodic web personalization.

7.5 System Overview

Figure 7.2 shows the proposed system for providing periodic-based personalized services on the Semantic Web. The *Web Server* of a website records and stores all requests from users into the *Semantic Web Usage Logs*. In the proposed system, the personalized service provided by *Service Provider* can be invoked by *Service Requester* as follows:

- *Service Provider*. *Automatic Web Usage Ontology Generation* process generates and maintains the *Personal Web Usage Ontologies* of all registered users based on the *Semantic Web Usage Logs*, which are represented in OWL [MH04]. As such, the knowledge on the periodic access behavior of each user is shareable over the Semantic Web. To provide the personalized service, *Personalized Resources Generation Service* process generates a list of ranked personalized resources that the specific user is most likely interested in during a given period based on the corresponding service request and

Figure 7.3: The proposed approach for the automatic generation of Personal Web Usage Ontology.

Personal Web Usage Ontology. *Service Specification Creation* process creates Service Model, Service Grounding and Service Profile for the proposed personalized service in OWL-S [MBH⁺04].

- *Service Broker*. It registers the personalized service provided by *Service Provider*, such that the service can be discovered by *Service Requesters*.
- *Service Requester*. It could be the *Web Server* or *Other Web Servers* of websites in the similar domain. According to the service profile discovered from *Service Broker*, *Service Requester* sends a service request with a userID and a period condition to *Service Provider*. Then, it can prepare and deliver appropriate resources to the user according to the returned personalized resources.

In this research, we only focus on the implementation of the *Service Provider*, which includes *Automatic Web Usage Ontology Generation*, *Personalized Resources Generation Service* and *Service Specification Creation*.

7.6 Automatic Web Usage Ontology Generation

In this section, we propose an approach for automatic generation of Personal Web Usage Ontologies of individual users from semantic web usage logs. Figure 7.3 shows the proposed approach, which consists of the following major processes: Global Web Usage Lattice Construction, Personal Web Usage Lattice Construction and Web Usage Ontology Generation.

7.6.1 Global Web Usage Lattice Construction

The Global Web Usage Lattice Construction process consists of the following two steps:

Table 7.1: An example of a semantic web usage log.

UserID	Timestamp	URL	FeatureVector
User1	21/May/2005 08:20:01	URL1	#Concept1, #Concept2, #Concept3, ...
User1	21/May/2005 08:22:32	URL2	#Concept7, #Concept3, #Concept5, ...
User2	21/May/2005 08:22:50	URL7	#Concept1, #Concept3, #Concept8, ...
User1	21/May/2005 08:27:30	URL3	#Concept3, #Concept1, ...
User1	21/May/2005 09:10:02	URL5	#Concept7, #Concept8, #Concept3, ...
User2	21/May/2005 09:24:34	URL4	#Concept3, #Concept7, ...

- *Identifying Attributes.* This step identifies a set of *periodic attributes* (i.e., temporal concepts such as morning and evening) and a set of *resource attributes* (i.e., useful concepts on domain ontology) to represent periodic-based web access activities according to the original semantic web usage logs and the mining purposes.
- *Constructing Global Web Usage Lattice.* From the predefined periodic and resource attributes, this step constructs the Global Web Usage Lattice to represent all periodic-based *global web access activities* and the hierarchical relationships between these activities.

Identifying Attributes

In the Semantic Web environment, web usage logs can be semantically enriched by associating each requested URL with one or more ontological entities such as concepts, attributes and relations to better describe the patterns of web navigation. Currently, there is no standard format for semantic web usage logs. Oberle *et al.* [OBHG03] proposed a framework for semantic enrichment of web usage logs. An example of a semantic web usage log is given in Table 7.1. As shown in Table 7.1, the semantic web usage log contains attributes on user identification (UserID), timestamp, requested URLs and feature vectors (FeatureVector). The user ID can be stored for non-anonymous user access sessions. The feature vectors indicate the ontological features that are presented in the web page of the corresponding URL. The format of such semantic web usage logs is similar to that of web usage logs used in Chapter 5, which was given in Table 5.2.

Note that the feature vector of each entry in semantic web usage logs may sometimes contain redundant information. For example, the feature vector of a URL may have feature concepts on “#DataMining” and “#AssociationRuleMining”, in which the latter is a sub-concept of the former. Since the subclass represents more specific concepts and inherits all

properties of its superclass, we may consider only the subclass features in the feature vectors.

As mentioned in Chapter 5, it is appropriate to use a set of *periodic* and *resource* attributes to represent personal web access activities. In this research, we have defined eight real-life temporal concepts, namely *Early Morning*, *Morning*, *Noon*, *Early Afternoon*, *Late Afternoon*, *Evening*, *Night* and *Late Night* as periodic attributes for web access activities. All ontological features contained in semantic web usage logs can be regarded as resource attributes for describing web access activities. However, not all features are necessary for some mining purposes. To cater for certain mining purposes, we can select only those important features as the resource attributes. The selection of the appropriate domain-specific ontological features depends on the specific applications. For example, the ontological features of product catalogs can be used for an e-commerce's website, whereas the ontological features of animals can be used for a zoo's website.

Constructing Global Web Usage Lattice

We represent a web access activity using a set of selected periodic attributes M_p and resource attributes M_r .

Definition 7.1. Let $|M_p| = a$ and $|M_r| = b$ be the total numbers of selected periodic attributes and resource attributes respectively. A set $B_k \subseteq M_p \cup M_r$ with i periodic attributes ($1 \leq i \leq a$) and j resource attributes ($1 \leq j \leq b$) is defined as a *global web access activity* of all users. In addition, $B_0 = \emptyset$ is defined as a virtual global web access activity.

$W_G = \{B_k\}$ denote the set of all global web access activities. $|W_G|$ is the total number of global web access activities. As such, there should be a total of $\binom{a}{i} \times \binom{b}{j}$ global web access activities with i periodic attributes and j resource attributes, where $\binom{a}{i}$ and $\binom{b}{j}$ denote the number of combinations. Therefore, $|W_G| = \sum_{i=1}^a \sum_{j=1}^b \binom{a}{i} \times \binom{b}{j} + 1$.

Definition 7.2. For two global web access activities $B_i, B_j \in W_G$ ($i \neq j$), B_i is a *sub-activity* of B_j , denoted as $B_i <_{W_G} B_j$, if and only if $B_j \subset B_i$, since B_i can represent more specific periodic-based web access patterns than B_j . Equivalently, B_j is a *super-activity* of B_i . $<_{W_G}$ is a partial order on W_G , called *global activity relationship*. In particular, if $B_i <_{W_G} B_j$ and there is no $B_k \in W_G$ ($B_k \neq B_i$ and $B_k \neq B_j$) such that $B_i <_{W_G} B_k <_{W_G} B_j$, then B_i is a *direct sub-activity* of B_j , and B_j is a *direct super-activity* of B_i . We denote this as $B_i \prec_{W_G} B_j$. $\prec_{W_G}$ is called a *direct activity relationship*.

Obviously, the virtual global web access activity $B_0 = \emptyset$ is the super-activity of all other global web access activities.

Figure 7.4: An example of the Global Web Usage Lattice.

For each global web access activity with i periodic attributes ($1 \leq i \leq a$) and j resource attributes ($1 \leq j \leq b$), it has $(a-i) + (b-j)$ direct sub-activities. And the virtual activity w_0 has $a \times b$ direct sub-activities. Then, a total of $a \times b + \sum_{i=1}^a \sum_{j=1}^b \{((a-i) \times (b-j)) \times [(a-i) + (b-j)]\}$ direct sub-activity relationships $\prec_{W_G}$ exist among all global web access activities. Using all direct sub-activity relationships, we can construct an entire lattice, called *Global Web Usage Lattice*, of all global web access activities of users.

Definition 7.3. A *Global Web Usage Lattice* based on the sets of periodic attributes (M_p) and resource attributes (M_r) is $L_G = (W_G, \prec_{W_G})$, where W_G is the set of all global web access activities, and $\prec_{W_G}$ is a partial order on W_G to represent the hierarchical relationships of all global web access activities.

Figure 7.4 shows an example that contains three periodic attributes “P1 (Late Afternoon)”, “P2 (Evening)” and “P3 (Night)”, and three resource attributes “R1 (Sports concept)”, “R2 (Games concept)” and “R3 (Chat concept)”. There are a total of 50 nodes representing global web access activities with a total of 135 edges representing direct sub-activity relationships.

7.6.2 Personal Web Usage Lattice Construction

The Global Web Usage Lattice constructed is huge since it contains a large number of global web access activities to cover all possible user web access activities. In general, the number of web access activities of an individual user is much smaller than the number of global web access activities. In this step, we attempt to identify a user’s web access activities and construct a Personal Web Usage Lattice from the access sessions of the user, which is

just a small sub-lattice with fuzzy logic representation of the Global Web Usage Lattice. The detailed steps for constructing the Personal Web Usage Lattice have been discussed in Chapter 5.

7.6.3 Web Usage Ontology Generation

In this process, we build the Web Usage Ontology for storing and sharing individual users' periodic-based web access activities for the Semantic Web. The generated Web Usage Ontology is represented using OWL (Web Ontology Language) [MH04]. The Web Usage Ontology Generation process consists of the following two steps:

- *Generating Global Web Usage Ontology.* This step generates the Global Web Usage Ontology from the Global Web Usage Lattice by mapping global web access activities and their hierarchical relationships into activity classes and their properties.
- *Generating Personal Web Usage Ontology.* This step generates the Personal Web Usage Ontology for a user from the Personal Web Usage Lattice and the Global Web Usage ontology. The Personal Web Usage Lattice, which represents the personal web access activity hierarchy, is a sub-lattice of the Global Web Usage Lattice. The Personal Web Usage Ontology is generated by mapping the personal web access activities in the Personal Web Usage Lattice into activity instances of the corresponding activity classes in the Global Web Usage Ontology.

Generating Global Web Usage Ontology

In the Semantic Web, ontology provides a common and shared knowledge of certain domains. Typically, ontology consists of a taxonomy with a set of inference rules. And the taxonomy can be expressed as a set of domain concepts (i.e., classes of objects) and the relationships among them (i.e., class hierarchy). Based on the formal definition of ontology [BEH⁺02], we define the Global Web Usage Ontology as follows.

Definition 7.4. A Global Web Usage Ontology is $O_G = (C, <_C, Q)$, where

- C is a set of activity classes (or concepts).
- $<_C$ is a partial order on C , called activity class hierarchy or taxonomy. If $c_i <_C c_j$, for $c_i, c_j \in C$ ($i \neq j$), then c_i is a sub-activity class of c_j , and c_j is a super-activity class of c_i . If there is no $c_k \in C$ ($k \neq i$ and $k \neq j$) with $c_i <_C c_k <_C c_j$, then c_i is a direct sub-activity class of c_j , and c_j is a direct super-activity class of c_i . This is denoted as $c_i \prec_C c_j$.

- Q is a set of properties (or relations) which consists of attribute properties $Q_A = \{q_i\}$ to represent periodic and resource attributes of activity classes, one taxonomy property q_T to represent the direct sub-activity relationship between two activity instances, and two quality properties, q_{sup} and q_{conf} , to represent the values of support and confidence of each activity instance.

Global Web Usage Ontology can be generated through *class and hierarchy mapping*, and *property mapping*. Class and hierarchy mapping initializes a set of activity classes C and builds the activity class hierarchy $<_C$ according to the Global Web Usage Lattice. Property mapping generates a set of properties Q , and assigns them to the corresponding activity classes by setting the domains of the properties.

The class and hierarchy mapping is performed in the following two steps:

- *Class mapping.* All global web access activities $B_i \in W_G$ ($0 \leq i \leq |W_G|$) in the Global Web Usage Lattice are mapped into activity classes $c_i \in C$ in the Global Web Usage Ontology. The name (class ID) of each activity class $c_i \in C$ ($i > 0$) is given based on the attributes involved in its corresponding global web access activity with “Activity_” as the prefix. For example, the activity class based on the global web access activity $\{P2, R3\}$ in Figure 7.4 is named as “Activity_Evening_Chat”. For the activity class c_0 based on the virtual activity $B_0 \in W_G$, we name it as “WebAccessActivity”.
- *Hierarchy mapping.* The activity hierarchy $<_{W_G}$ in the Global Web Usage Lattice is mapped into the activity class hierarchy $<_C$ in the Global Web Usage Ontology. If $B_i <_{W_G} B_j$, for $B_i, B_j \in W_G$ ($i \neq j$), then there exists $c_i <_C c_j$, for $c_i, c_j \in C$ ($i \neq j$). As such, the activity class c_0 is the root class in the activity class hierarchy. All other activity classes are its (direct or indirect) sub-activity classes and inherit its properties.

The activity class based on the global web access activity $\{P2, R3\}$ in Figure 7.4, is defined as

```
<owl:Class rdf:ID="Activity_Evening_Chat">
  <rdfs:subClassOf rdf:resource="#WebAccessActivity"/>
</owl:Class>
```

whereas the activity class based on the global web access activity $\{P2, R1, R3\}$ is defined as

```
<owl:Class rdf:ID="Activity_Evening_Sports_Chat">
  <rdfs:subClassOf rdf:resource="#Activity_Evening_Sports"/>
```

```
<rdfs:subClassOf rdf:resource="#Activity_Evening_Chat"/>
</owl:Class>
```

Note that only direct super-activity classes are declared in the “partial” part in the definition of each activity class, as all indirect super-activity class relationships can be derived from direct super-activity class relationships. For example, the activity class “WebAccessActivity” is not included in the “partial” part of the activity class “Activity_Evening_Sports_Chat”.

The property mapping comprises *attribute property mapping*, *taxonomy property mapping* and *quality property mapping*:

- *Attribute Property Mapping.* All periodic and resource attributes $m_i \in M_p \cup M_r$ ($0 < i \leq |M_p \cup M_r|$) are mapped into attribute properties $q_i(X_i, [0, 1]) \in Q_A$, where $X_i = \{c_k \mid c_k \in C, \text{ and } m_i \in B_k, B_k \in W_G \text{ is the corresponding global web access activity of } c_k.\}$. X_i is the domain of q_i , which is defined as the set of activity classes whose corresponding global web access activities involve the attributes m_i . $[0, 1]$ indicates the range of q_i , which is a fuzzy membership value in $[0, 1]$. The name of each attribute property is given according to its corresponding attribute with “during” as the prefix of periodic attributes or “access” as the prefix of resource attributes. For example, the property based on the periodic attribute P2 is named as “duringEvening”, and the property based on the resource attribute R3 is named as “accessChat”. The attribute property “duringEvening” is defined as

```
<owl:DatatypeProperty rdf:ID="duringEvening">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Activity_Evening_Sports"/>
        <owl:Class rdf:about="#Activity_Evening_Games"/>
        <owl:Class rdf:about="#Activity_Evening_Chat"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="&xsd;float"/>
</owl:DatatypeProperty>
```

whereas the attribute property “accessChat” is defined as

```
<owl:DatatypeProperty rdf:ID="accessChat">
  <rdfs:domain>
    <owl:Class>
```

```

    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#Activity_LateAfternoon_Chat"/>
      <owl:Class rdf:about="#Activity_Evening_Chat"/>
      <owl:Class rdf:about="#Activity_Night_Chat"/>
    </owl:unionOf>
  </owl:Class>
</rdfs:domain>
<rdfs:range rdf:resource="#xsd:float"/>
</owl:DatatypeProperty>

```

Note that only activity classes $c_k \in X_i$ with no other super-activity class $c_j \in X_i$ are declared in the “domain” part in the definition of each attribute property, as all sub-activity classes of c_k can inherit the corresponding attribute property. For example, as the activity class “Activity_Evening_Sports_Chat” is the subclass of “Activity_Evening_Sports” and “Activity_Evening_Chat”, it is not included in the “domain” part of the attribute property “duringEvening”.

- *Taxonomy Property Mapping.* One taxonomy property $q_T = \text{hasDirectSubActivity}(C, C)$ is created to represent the direct sub-activity relationship between two activity instances. The domain and range of q_T are defined as C , i.e., the set of all activity classes. The taxonomy property $\text{hasDirectSubActivity}(C, C)$ is defined as

```

<owl:ObjectProperty rdf:ID="hasDirectSubActivity">
  <rdfs:domain rdf:resource="#WebAccessActivity"/>
  <rdfs:range rdf:resource="#WebAccessActivity"/>
</owl:ObjectProperty>

```

Note that only the activity class “WebAccessActivity” is set with domain and range, as all other activity classes are its direct or indirect sub-activity classes that can inherit the $\text{hasDirectSubActivity}(C, C)$ property.

- *Quality Property Mapping.* Two quality properties $q_{sup} = \text{hasSupport}(C, [0, 1])$ and $q_{conf} = \text{hasConfidence}(C, [0, 1])$ are created to represent values of support and confidence of activity instances. The quality property “hasSupport” is defined as

```

<owl:ObjectProperty rdf:ID="hasSupport">
  <rdfs:domain rdf:resource="#WebAccessActivity"/>
  <rdfs:range rdf:resource="#xsd:float"/>
</owl:ObjectProperty>

```

whereas the quality property “hasConfidence” is defined as

Figure 7.5: Representing an activity class in OWL.

```

<owl:ObjectProperty rdf:ID="hasConfidence">
  <rdfs:domain rdf:resource="#WebAccessActivity"/>
  <rdfs:range rdf:resource="#xsd:float"/>
</owl:ObjectProperty>

```

The Global Web Usage Ontology $O_G = (C, <_C, Q)$ can be generated after the automatic class and hierarchy mapping, and property mapping processes. Figure 7.5 shows the OWL representation of the activity class “Activity_Evening_Chat” in the Global Web Usage Ontology which is mapped from the global web access activity $\{P2, R3\}$ in the Global Web Usage Lattice. For the example Global Web Usage Lattice given in Figure 7.4, a Global Web Usage Ontology can be generated automatically with a total of 50 activity classes.

OWL allows extension of imported definitions without the need of modifying the original ontology and also supports incremental construction of ontology, which makes it easy for incremental update of the activity class hierarchy. When new periodic attributes or resource attributes are introduced, we just need to create new activity classes with new attributes and insert new subclass relationships between the new and existing activity classes to extend the activity class hierarchy.

Generating Personal Web Usage Ontology

We define Personal Web Usage Ontology as follows.

Definition 7.5. A Personal Web Usage Ontology of a user is $O_P = (O_G, I_P, <_{I_P})$, where

- $O_G = (C, <_C, Q)$ is the Global Web Usage Ontology.
- I_P is a set of activity instances of the corresponding activity classes in C .
- $<_{I_P}$ is a partial order on I_P , called activity instance hierarchy or taxonomy. If $i_i <_{I_P} i_j$, for $i_i, i_j \in I_P$ ($i \neq j$), then i_i is a sub-activity instance of i_j , and i_j is a super-activity instance of i_i . If there is no $i_k \in I_P$ ($k \neq i$ and $k \neq j$) with $i_i <_{I_P} i_k <_{I_P} i_j$, then i_i is a direct sub-activity instance of i_j , and i_j is a direct super-activity instance of i_i . This is denoted as $i_i \prec_{I_P} i_j$.

To generate Personal Web Usage Ontology, we combine the Personal Web Usage Lattice of a user with the Global Web Usage Ontology using *instance mapping*. Instance mapping generates a set of activity instances I_P from the corresponding activity classes in the Global Web Usage Ontology, and an activity instance hierarchy $<_{I_P}$.

Instance mapping is carried out as follows. All personal web access activities $v(B_i) \in W_P$ ($0 \leq i \leq |W_P|$ and $B_i \in W_G$) in the Personal Web Usage Lattice are mapped into the activity instance i_i of activity class $c_i \in C$ based on the global web access activity $B_i \in W_G$. The name (instance ID) of each activity instance is given based on its corresponding activity class with its user ID as the suffix. For example, the activity instance based on the personal web access activity $\{P2:0.8, R3:0.9\}$ of a user called “User1” given in Figure 5.5 is named as “Activity_Evening_Chat_User1”. The activity instance i_0 based on the virtual activity $v(\emptyset) \in W_P$ is named as “WebAccessActivity_User1”. The fuzzy membership values of attributes of a personal web access activity are mapped into the values of attribute properties of its corresponding activity instance. The values of two quality properties p_{sup} and p_{conf} of each activity instance are set to the values of fuzzy support and confidence of the corresponding personal web access activity.

The activity hierarchy $<_{W_P}$ in the Personal Web Usage Lattice is mapped into activity instance hierarchy in the Personal Web Usage Ontology. If $v(B_i) <_{W_P} v(B_j)$, for $v(B_i), v(B_j) \in W_P$ ($i \neq j$), then there exist $i_i <_{I_P} i_j$, for $i_i, i_j \in I_P$ ($i \neq j$). As such, activity instance i_0 is the root instance in the activity instance hierarchy. All other activity instances are its (direct or indirect) sub-activity instances. If personal web access activities $v(B_i) \prec_{W_P} v(B_j)$, then a taxonomy property $\text{hasDirectSubActivity}(i_j, i_i)$ is set in the activity instance $v(B_j)$.

Figure 7.6: Representing an activity instance in OWL.

For example, the activity instance based on the personal web access activity {P2:0.8, R3:0.9} of User1 with $Sup = 0.21$ and $Conf = 0.62$ in Figure 5.5 is created as

```
<Activity_Evening_Chat rdf:ID="Activity_Evening_Chat_User1">
  <hasDirectSubActivity
    rdf:resource="#Activity_Evening_Sports_Chat_User1"/>
  <duringEvening rdf:datatype="&xsd;float">0.8</duringEvening>
  <accessChat rdf:datatype="&xsd;float">0.9</accessChat>
  <hasSupport rdf:datatype="&xsd;float">0.21</hasSupport>
  <hasConfidence rdf:datatype="&xsd;float">0.62</hasConfidence>
</Activity_Evening_Chat>
```

Figure 7.6 shows the OWL representation of the personal web access activity {P2:0.8, R3:0.9} of User1 with $Sup = 0.21$ and $Conf = 0.62$ in the activity instance “Activity_Evening_Chat_User1” of the Personal Web Usage Ontology. The OWL definition of the corresponding activity class “Activity_Evening_Chat” has been given in Figure 7.5.

In general, the number of activity instances in the Personal Web Usage Ontology of a user is much less than the number of activity classes in the Global Web Usage Ontology. For our example, there are only 9 activity instances for a total of 50 activity classes.

In practice, we do not need to generate the complete Global Web Usage Ontology in advance, but only define the activity class when we need to generate its corresponding activity instance. Therefore, the construction of the Global Web Usage Ontology could be done incrementally and incorporated into the process of constructing the Personal Web Usage Ontology instead of an individual process. In practical situations, the incrementally generated Global Web Usage Ontology should be much smaller than the complete one. As such, it will only have a minor effect on the performance of the proposed approach.

Algorithm 7.1 PR_Gen_Service(uid, ts, te)**Input:** uid - a user ID $ts \in [0, 24]$ - a start time of a period condition $te \in [0, 24]$ - an end time of a period condition**Output:** PR_o - an ordered set of personalized resources**Process:**

```

1: Initialize  $PR_o \leftarrow \{\emptyset\}$ 
2: Search for the target Personal Web Usage Ontology  $O_P$ , whose UserID property is  $uid$ 
3: if  $O_P \neq null$  then
4:   if  $ts \leq te$  then
5:      $p_c = [ts, te]$ 
6:   else
7:      $p_c = [0, te] \cup [ts, 24]$ 
8:   end if
9:    $p \Rightarrow PR_o(p_c, O_P) \leftarrow \mathbf{PR\_Gen}(p_c, O_P)$ 
10:   $PR_o \leftarrow PR_o(p_c, O_P)$ 
11: end if
12: return  $PR_o$ 

```

7.7 Personalized Resources Generation Service

In this section, we attempt to incorporate the periodic web personalization approach proposed in Chapter 6 into the *Personalized Resources Generation Service*.

The *Personalized Resources Generation Service* can be implemented based on Algorithm 7.1. Since the Personal Web Usage Ontology has the same structure as the Personal Web Usage Lattice proposed in Chapter 5, the Algorithm $\mathbf{PR_Gen}(p_c, O_P)$ can be obtained easily by modifying Algorithm 6.1 given in Chapter 6. The start and end times of the input period condition depend on the practical web services. The users can define them according to their real practical requirements. For example, a news website could set $ts = 7.0$ and $te = 9.0$ to provide personalized morning news to registered users between 7 a.m. and 9 a.m.. For another example, a web forum could set users' request time as ts and te to provide personalized topic or article list to users when they visit the website.

7.8 Service Specification Creation

In this section, we create the service specification of the proposed *Personalized Resources Generation Service* in OWL-S for publishing the periodic-based personalized service on the Semantic Web environment as a kind of Semantic Web services.

Figure 7.7: Inputs and outputs of the PersonalizedResourcesGeneration process.

7.8.1 Creating the Service Model

To create the Service Model, we start with modeling the services as processes. We describe the main function for our periodic-based personalized service on the Semantic Web, called `PersonalizedResourcesGeneration`. It will be modeled as an atomic process, as we do not intend to further refine it. Then, we need to decide what are its inputs and outputs. Figure 7.7 shows the inputs and outputs for the atomic process “`PersonalizedResourcesGeneration`”. The name with a type (in brackets) is specified for each input and output.

The process “`PersonalizedResourcesGeneration`” is defined as follows:

```

<process:AtomicProcess rdf:ID="PersonalizedResourcesGeneration">
  <process:hasInput rdf:resource="UserID"/>
  <process:hasInput rdf:resource="PeriodStartTime"/>
  <process:hasInput rdf:resource="PeriodEndTime"/>
  <process:hasOutput rdf:resource="PersonalizedResources"/>
</process:AtomicProcess>

<process:Input rdf:ID="UserID">
  <process:parameterType rdf:resource="xsd:string"/>
</process:Input>

<process:Input rdf:ID="PeriodStartTime">
  <process:parameterType rdf:resource="xsd:time"/>
</process:Input>

<process:Input rdf:ID="PeriodEndTime">
  <process:parameterType rdf:resource="xsd:time"/>
</process:Input>

<process:Output rdf:ID="PersonalizedResources">
  <process:parameterType rdf:resource="#ListOfResources"/>
</process:Output>
  
```


The parameter type of the output “PersonalizedResources”, i.e., the OWL class “ListOfResources”, is defined as follows:

```
<owl:Class rdf:ID="ListOfResources"/>

<owl:Class rdf:ID="Resource"/>

<owl:ObjectProperty rdf:ID="hasResource">
  <rdfs:domain rdf:resource="#ListOfResources"/>
  <rdfs:range rdf:resource="#Resource"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="hasPriority">
  <rdfs:domain rdf:resource="#Resource"/>
  <rdfs:range rdf:resource="#xsd:positiveInteger"/>
</owl:DatatypeProperty>
```

According to the above definitions, the instance of the class “ListOfResources” will consist of a list of instances of the class “Resource” through the data type property “hasResource”, and each instance of the class “Resource” has a positive integer as its priority. We use the position of each resource in the resource list as its priority, i.e., resource a has higher priority than resource b , if a occurs before b in the list of resources. For our example, we assume that three classes, i.e., “Sports”, “Games” and “Chat”, which are the subclasses of “Resource”, are defined as follows:

```
<owl:Class rdf:ID="Sports">
  <rdfs:subClassOf rdf:resource="#Resources"/>
</owl:Class>

<owl:Class rdf:ID="Games">
  <rdfs:subClassOf rdf:resource="#Resources"/>
</owl:Class>

<owl:Class rdf:ID="Chat">
  <rdfs:subClassOf rdf:resource="#Resources"/>
</owl:Class>
```

7.8.2 Creating the Service Grounding

The grounding of a service specifies the details of how to access the service, including protocol and message formats, serialization, transportation, and addressing. In OWL-S, both the Service Profile and the Service Model are regarded as abstract representations; only the Service

Grounding deals with the concrete level of specification. A grounding can be regarded as a mapping from an abstract to a concrete specification of those service description elements that are required for interacting with the service, especially including the inputs and outputs of atomic processes. It is easy to ground an OWL-S atomic process using the extensible elements provided by WSDL (Web Services Description Language), which describes network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information in XML format. An OWL-S atomic process with both inputs and outputs corresponds to a WSDL request-response operation. The service grounding of “PersonalizedResourcesGeneration” is defined below, where the WSDL input/output parameters of “#WSDL_uid”, “#WSDL_ts”, “#WSDL_te” and “#WSDL_PR” are defined in the corresponding WSDL document according to the parameters of uid , ts , te and PR_o in Algorithm 7.1, respectively.

```
<grounding:WsdgGrounding
  rdf:ID="PersonalizedResourcesGenerationService_Grounding">
  <service:supportedBy
    rdf:resource="#PersonalizedResourcesGeneration_Service"/>
  <grounding:hasAtomicProcessGrounding
    rdf:resource="#PersonalizedResourcesGeneration_Grounding"/>
</grounding:WsdgGrounding>

<grounding:WsdgAtomicProcessGrounding
  rdf:ID="PersonalizedResourcesGeneration_Grounding">
  <grounding:owlsProcess rdf:resource="#PersonalizedResourcesGeneration"/>
  <grounding:wsdlOperation>
    <grounding:WsdgOperationRef>
      <grounding:portType>
        <xsd:anyURI
          rdf:value="#PersonalizedResourcesGeneration_WSDL_PortType"/>
        </grounding:portType>
      <grounding:operation>
        <xsd:anyURI
          rdf:value="#PersonalizedResourcesGeneration_WSDL_Operation"/>
        </grounding:operation>
      </grounding:WsdgOperationRef>
    </grounding:wsdlOperation>
    <grounding:wsdlInputMessage>
      <xsd:anyURI rdf:value="#PersonalizedResourcesGeneration_WSDL_Input"/>
    </grounding:wsdlInputMessage>
    <grounding:wsdlInputs rdf:parseType="Collection">
      <grounding:WsdgInputMessageMap>
        <grounding:owlsParameter rdf:resource="#UserID"/>
        <grounding:wsdlMessagePart>
```

```

        <xsd:anyURI rdf:value="#WSDL_uid"/>
    </grounding:wsdlMessagePart>
</grounding:WsdInputMessageMap>
<grounding:WsdInputMessageMap>
    <grounding:owlsParameter rdf:resource="#PeriodStartTime"/>
    <grounding:wsdlMessagePart>
        <xsd:anyURI rdf:value="#WSDL_ts"/>
    </grounding:wsdlMessagePart>
</grounding:WsdInputMessageMap>
<grounding:WsdInputMessageMap>
    <grounding:owlsParameter rdf:resource="#PeriodEndTime"/>
    <grounding:wsdlMessagePart>
        <xsd:anyURI rdf:value="#WSDL_te"/>
    </grounding:wsdlMessagePart>
</grounding:WsdInputMessageMap>
</grounding:wsdlInputs>
<grounding:wsdlOutputMessage>
    <xsd:anyURI rdf:value="#PersonalizedResourcesGeneration_WSDL_Output"/>
</grounding:wsdlOutputMessage>
<grounding:wsdlOutputs rdf:parseType="Collection">
    <grounding:WsdOutputMessageMap>
        <grounding:owlsParameter rdf:resource="#ListOfResources"/>
        <grounding:wsdlMessagePart>
            <xsd:anyURI rdf:value="#WSDL_PR"/>
        </grounding:wsdlMessagePart>
    </grounding:WsdOutputMessageMap>
</grounding:wsdlOutputs>
</grounding:WsdAtomicProcessGrounding>

```

7.8.3 Creating the Service Profile

OWL-S has been designed to enable automatic web service discovery by providing descriptions of the properties and capabilities. These descriptors exist in a registry of services to provide better indexing and retrieval features by the agent for matchmaking of web services [BV03]. For our example, the instances of the classes “Service” and “Profile” are created as follows:

```

<profile:Profile rdf:ID="PersonalizedResourcesGeneration_Profile">
    <profile:serviceName rdf:datatype="&xsd:string">
        Personalized_Resources_Generation_Service
    </profile:serviceName>
    <profile:textDescription rdf:datatype="&xsd:string">
        An online service to provide a list of ordered personalized resources
        based on a UserID and a period condition.
    </profile:textDescription>
    <profile:hasParameter rdf:resource="#UserID"/>
    <profile:hasParameter rdf:resource="#PeriodStartTime"/>
    <profile:hasParameter rdf:resource="#PeriodEndTime"/>

```

```

    <profile:hasParameter rdf:resource="#ListOfResources"/>
    <profile:hasInput rdf:resource="#UserID"/>
    <profile:hasInput rdf:resource="#PeriodStartTime"/>
    <profile:hasInput rdf:resource="#PeriodEndTime"/>
    <profile:hasOutput rdf:resource="#ListOfResources"/>
    <service:presentedBy
      rdf:resource="#PersonalizedResourcesGeneration_Service"/>
  </profile:Profile>

  <service:Service rdf:ID="PersonalizedResourcesGeneration_Service">
    <service:presents
      rdf:resource="#PersonalizedResourcesGeneration_Profile"/>
    <service:describedBy rdf:resource="#PersonalizedResourcesGeneration"/>
    <service:supports
      rdf:resource="#PersonalizedResourcesGeneration_Grounding"/>
    <service:providedBy rdf:resource="http://www.MyWeb.com"/>
  </service:Service>

```

7.9 Prototype Service Requester System

We have developed a prototype Service Requester System aiming for demonstrating the applicability of the proposed Semantic Web service for personalized resources generation. Figure 7.8 shows the interface of the prototype Service Requester System. According to the user's inputs, i.e., *UserID*, *PeriodStartTime* and *PeriodEndTime*, the web personalized service returns a ranked list of personalized resources with the information of the corresponding period-supported activities based on the Personal Web Usage Ontology of the specified user. This prototype Service Requester System has illustrated that the knowledge of personalized resources can be derived from the sharable Web Usage Ontology successfully through the proposed Semantic Web personalized service. Such knowledge is very useful for supporting periodic web personalization on the Semantic Web. According to the ranked list of personalized resources, the web service providers on the Semantic Web can deliver related information, such as news, books, advertisements, product catalog and other recommended content, to the user during the specific period.

7.10 Summary

In this chapter, we have proposed a semantic web usage mining approach for generating the Personal Web Usage Ontology automatically based on the Personal Web Usage Lattice model proposed in Chapter 5. The Global Web Usage Lattice is constructed for describing all

Figure 7.8: Prototype Service Requester System for personalized resources generation service.

possible web access activities, and then, used to construct the Global Web Usage Ontology using OWL. Each personal web access activity in the Personal Web Usage Lattice is mapped into one instance of the corresponding activity class and its properties of the Global Web Usage Ontology. The generated Personal Web Usage Ontology has the same structure as the Personal Web Usage Lattice. In addition, we have also incorporated the periodic web personalization approach proposed in Chapter 6 into the Semantic Web environment using the technologies of Semantic Web services. The prototype Service Requester System has also been developed to show that the proposed periodic-based personalized web service can be invoked successfully by a Service Requester.

Chapter 8

Conclusions

8.1 Summary

With the explosive growth of information available on the World Wide Web, it has become much more difficult to access relevant information from the Web. One possible solution to solve this problem is web personalization, which aims to customize the content and structure of a website to the needs of specific users taking advantage of the knowledge acquired from the analysis of the users' access behaviors. To capture users' web access behaviors, web usage mining, which discovers interesting and frequent users' access patterns from web usage logs, is one of the most promising techniques.

To achieve effective web personalization on the Web and the Semantic Web, advanced techniques for pattern discovery from web usage logs, pattern matching and ontology generation have been investigated. The major objectives that have been achieved in this research are summarized as follows:

- An efficient sequential access pattern mining algorithm, called CSB-mine (Conditional Sequence Base mining algorithm), has been developed to discover sequential access patterns from web usage logs. The proposed CSB-mine algorithm is based directly on the conditional sequence bases of each frequent event, which eliminates the need for the construction of the initial WAP-tree and the re-construction of costly intermediate conditional WAP-trees as in the conventional WAP-mine algorithm. Experimental results have shown that the proposed CSB-mine algorithm has performed much more efficient than the WAP-mine algorithm, especially when the support threshold becomes smaller and the size of database gets larger.
- The Sequential Web Access based Recommender System (*SWARS*) has been developed

to achieve effective web recommendations. In the proposed *SWARS*, the CSB-mine algorithm is used to mine sequential access patterns. The mined patterns are stored in a compact tree structure, called Pattern-tree, which is then used for efficient user access pattern matching and generating web links for online recommendations. The experimental results have shown that the proposed web recommender system is very effective for recommending related web pages to a user that are most probably to be accessed next or in the near future by the user.

- A fuzzy Formal Concept Analysis (FCA) based web usage mining approach has been proposed for discovering a specific kind of periodic association access patterns of individual users from semantic enriched web usage logs. The proposed approach uses fuzzy set theory to represent both real-life temporal concepts and meaningful requested resources, and incorporates them into FCA for constructing a novel user behavior model, called Personal Web Usage Lattice, from which periodic association access patterns can be extracted.
- A Personal Web Usage Lattice based approach has been proposed for achieving effective periodic web personalization. Different from non-periodic approaches, the proposed periodic web personalization approach can determine effectively which resources a user is most probably interested in during a given period based on the user's Personal Web Usage Lattice without the use of the user's current access information. This makes it possible to perform more costly personalized resource preparation in advance rather than in real-time. The experimental results have shown that the proposed approach has achieved very effective web personalization evaluated by the applicability and satisfaction measures for predefined periods. In addition, a comparable performance has been obtained in comparison with an association rule based non-periodic approach for online resource preparation purposes.
- A semantic web personalization framework has been developed. In the proposed approach, Web Usage Ontologies are generated based on the Global and Personal Web Usage Lattices. The proposed periodic web personalization approach has also been extended for supporting periodic-based personalized services on the Semantic Web.

8.2 Future Work

In this research, we have proposed advanced web usage mining techniques for discovering web access behavior patterns of users from web usage logs. We have also applied the discovered access patterns to support personalized web services, such as web recommendations and periodic web personalization. Moreover, in order to extend the proposed techniques to the Semantic Web environment, we have also proposed ontology generation techniques for generating Web Usage Ontology automatically based on our proposed user behavior model and Semantic Web services for supporting periodic web personalization on the Semantic Web. Apart from continuing further enhancement of the proposed research works, this research can also be further extended in the following directions: web user clustering, client-side web logs mining, FCA based web usage retrieval, extending the Semantic Web personalization approach and applying our techniques to other practical applications.

8.2.1 Further Enhancement of the Proposed Research Works

In this research, several novel approaches have been proposed to discover useful access patterns for supporting effective personalized web services. However, some specific issues still have to be resolved to further enhance the proposed approaches, which are listed as follows.

- *Dealing with noises in web usage data.* Web usage data often have irrelevant and noisy entries, such as requests of images, sessions of robots/crawlers, sessions from proxies, automatic requests of advertisement pages and so on. Therefore, before applying pattern discovery techniques, the original web usage logs usually need to be cleaned and reformatted. The main preprocessing tasks have been introduced in Section 2.2. In the proposed approaches, only simplified tasks are applied, because we use specific web usage data with little noises as our experimental datasets. For future work, we can enhance the preprocessing phase to handle noises in web usage data, which includes filtering out robot sessions and identifying more reasonable session boundaries.
- *Semantic annotation for web usage data.* In the periodic association access pattern mining approach proposed in Chapter 5, we assume that the annotation of web usage log with semantic information has been done by the designers or administrators of the websites manually or semi-automatically without going into details. Many approaches can be used for annotating web usage logs, such as automatic classification [LHF02]. In practice, semantic annotation of web usage logs is another research topic on its own [MLME04, FMM03, EVV03, OBHG03, DM02], which is a good topic for future work.

- *Investigating new measures for mining access patterns.* In this research, we only use the support and confidence thresholds to filter uninteresting access patterns, which are widely used in association rule mining approaches. Some of the interestingness measures available are interest [BMS97], conviction [BMUT97] and reliability [AEMT00]. These measures could be investigated for future extension.
- *Distributed and incremental approach.* Since the Web is a continuous and heterogeneous entity, an applicable web usage mining approach should be distributed and incremental. Web usage mining approaches need to be distributed, because web services are usually globally distributed and very diverse. Thus, the issue is how to integrate the various web usage data from different machines together. Web usage mining approaches need to be incremental, because we do not want to throw away all previous mining results when new data arrive. Thus, mining the web usage data incrementally becomes an important problem. For future work, we will consider the above issues for further extension of the proposed approaches.

8.2.2 Web User Clustering

In this research, we have proposed two web personalization approaches, which are based on a general user behavior model, i.e., Pattern-tree, and a personal user behavior model, i.e., Personal Web Usage Lattice, respectively. The general user behavior model treats all users as anonymous and stores access patterns that occur frequently in web usage logs of all users. Regardless of who visits the website, we assume that the user has similar frequent access behavior or main interests as that of all other users. The advantage is that we can provide personalized services for every user, even for new users without any access records before. However, such general user behavior model ignores the special access characteristics of individual users. Web services using such models may satisfy the needs of most users, but may not be able to cater for the needs of some users. The personal user behavior model only consists of access patterns of a particular user, which can be used for providing more effective personalized services. Nevertheless, generating and maintaining personal user behavior models for all identifiable users are very costly. Therefore, in order to keep a balance between the accuracy and the cost of personalized web services, we can build user group behavior models for groups of users instead of personal behavior models. Web user clustering aims to apply clustering techniques to discover groups of users with similar browsing preferences and habits from web usage logs. Then, all access records of users from the same

user cluster can be used for generating user behavior model supporting personalized web services of these users. For future work, we can investigate clustering techniques for web user clustering based on web usage logs. The basic idea is given as follows.

The proposed CSB-mine algorithm is an efficient sequential access pattern mining algorithm. Each discovered sequential access pattern is supported by a set of access sessions in web usage logs. Assume that all users can be identified by their unique user IDs. For each sequential access pattern, we can find a group of users who have a certain proportion of access sessions supporting that pattern. Each group of users can be regarded as an initial user cluster, and the corresponding sequential access pattern can be used as the cluster label. The objective of this future work is to develop a clustering algorithm to combine initial user clusters with similar access characteristics into final user clusters. Since each user may have multiple behaviors and interests, we allow that each user can belong to several clusters, i.e., some clusters may have overlapping. In addition, cluster labels of initial clusters, i.e., discovered sequential access patterns, need to be combined to form several Pattern-trees as labels of final user clusters.

8.2.3 Client-side Access Logs Mining for User Access Behavior

Most of the proposed web usage mining research focus mainly on analyzing web server logs. In fact, client-side web access logs (or browser logs) can record more accurate and comprehensive web access information of users, which is very useful for user access behavior analysis, and user profiling and modeling. As such, one of the future directions is to mine client-side access logs.

Client-side usage data can be collected using a specialized monitoring agent (e.g., Letizia [Lie95] and Ergotracer [RPG02]) installed at the client machine or a modified web browser (e.g. Web Browser Extended [LLS03]) with the capability of tracing any web activities of users. In addition, the network sniffing technique [Her04] is another approach for recording client-side web usage data from a network of users. The client-side usage data may contain single-user access activities or a group of user activities.

In this research, we have developed a specialized client monitoring program, which was developed based on Browser Helper Object (BHO) installed at the client machine to record the surfing activities of users. The recorded data is then stored in client-side logs. Then, the proposed periodic association access pattern mining approach discussed in Chapter 5 is applied to the client-side logs to mine interesting and frequent web access behavior patterns

and visualize them on the screen [ZHF05]. The visualization of web access behavior is particularly useful for users such as parents, managers or employers who would like to understand the access behavior of their children and employees.

The client-side logs generally consist of the following information: *UserID*, *StartTime*, *EndTime*, *RequestURL* and *CategoryID*. Compared with web server logs, the client-side logs have the following special properties:

- *Real user-based logs.* The client-side logs are based on individual users, but not on individual client machines. This is especially useful for environments such as the Intranet of a company and the local area network within a department where all users are usually assigned with a unique user name. All client machines' identities can be recorded together with the users' web access information to the client-side logs. As such, accurate behaviors of individual users can be mined and analyzed.
- *More accurate surfing information.* The client-side recording program can also capture the end time of a web page viewed by checking the status of the window (such as the closing of the window or loading of a new web page). This method is more accurate than using the time difference between two adjacent entries in the same access session to estimate the duration of a web page view. As such, a user's interest on a certain web page can be measured by its duration.
- *URL categorization.* The URLs in web logs contain little semantic information about the web contents accessed by users. To overcome this problem, each URL can be mapped into a predefined category or topic such as News, Sports and Entertainment. The category information can be obtained using a web page category classification technique [LHF02]. As such, each user access session can be represented as a sequence of categories instead of a sequence of individual URLs. This practical representation is very useful for understanding the preference and behavior of a user.
- *Multi-user logs.* The logs contain access patterns from all clients. Therefore, apart from mining the activities of an individual user, we can also mine the common interests and habits of a group of users, and find the outliers (users with unusual access behaviors).

Most of the current approaches have made use of the client-side usage data for web link recommendations. For future work, apart from mining user access patterns for web recommendations, we can also mine the client access patterns for analyzing various user behaviors, such as distinguishing users' long-term and temporary interests, extracting the common behaviors of a group of users and detecting the outliers. In addition, we can also

explore web usage mining techniques for user profiling [MS00b], which aims to learn users' habits, interests, patterns and preferences for practical applications.

8.2.4 FCA based Web Usage Retrieval

Formal Concept Analysis is a powerful technique for data analysis. We have proposed a framework on fuzzy FCA based web usage mining in Chapter 5. A Personal Web Usage Lattice is constructed from the original web server logs and then exploited for further mining process. As discussed in Chapter 5, Formal Concept Analysis can also be used as an unsupervised learning technique for conceptual clustering, and supporting efficient and effective information retrieval. In future work, we intend to investigate FCA based information retrieval from web usage data.

Related research work on text retrieval using concept lattice has been proposed. In [GMA93], a controlled experiment was conducted to compare the information retrieval method using the Galois lattice (the alias of concept lattice) with two other conventional retrieval methods, namely Boolean querying using index terms and navigation in a manually built hierarchical classification structure. The most attractive feature of the lattice-based retrieval method is its capability to combine retrieval and direct term queries within a unique concept hierarchical structure. The results have shown that there is no significant difference in performance between Boolean querying method and the Galois lattice retrieval method. However, hierarchical classification structure based retrieval has achieved significantly lower recall compared to the other two methods.

In [CR96], Carpineto and Romano presented a system for information exploration and retrieval through a specific lattice representation built from a text database. It also compared information retrieval using lattice-based hybrid navigation with conventional Boolean querying. The results have shown that the lattice-based retrieval has achieved comparable performance with Boolean retrieval on medium-sized databases.

In [KC01], Kim and Compton proposed a domain-specific document retrieval system using a browsing mechanism based on the concept lattice of Formal Concept Analysis in cooperation with an incremental knowledge acquisition mechanism. The concept lattice can be incrementally and automatically reformulated whenever a new document is added or the existing documents are changed. The concept lattice of FCA is a useful way of supporting the flexible management of documents required by individuals, small communities or in specialized domains.

All previous research works have shown that FCA based information retrieval methods are effective and efficient for textual data. Web usage data has similar characteristics to textual data. In textual data, a document can be represented by a set of keywords, whereas in web usage data, a session or a user can be represented by a set of resource attributes, e.g., URLs, categories or topics. Different combinations of keywords will give different topics of documents, while different combinations of resource attributes will indicate different interests of access sessions or users. In addition, web usage data contains more information than the textual data, such as the timestamp of each request, duration of each page view and so on. We can also extract additional information from the contents of relevant web pages for analysis. Therefore, Formal Concept Analysis is potentially an effective and efficient technique for web usage data retrieval.

However, there are some differences between textual data and web usage data. Firstly, requested URLs in web logs are less meaningful than keywords in documents. Mapping URLs to general concepts is difficult and expensive. For web server logs, the web designer may be able to provide a meaningful topic for each web page, but for most of the websites we have to convert URLs into topics or descriptions during data preprocessing. This is not an easy task. To do this, we may need to utilize some web content mining techniques. Moreover, since the Semantic Web provides more semantics for web content, it is a promising method to solve the above problem using the knowledge from the Semantic Web, such as ontology.

Secondly, web usage data is multi-level and multi-dimensional. There are different views of web usage data, such as a single URL (web page), URLs from one website, URL from one category, an access session, access sessions from one user, etc. Information retrieval from web usage data should provide browsing and searching capabilities from multi-level and multi-dimensional viewpoints. For example, a user may want to search for a certain category by a given URL first, and then search for users who are interested in that category. Another example is that a user may search for a set of categories in which the user is interested, and then searches for all access sessions with URLs belonging to these categories. Such queries must be performed upon several Web Usage Lattices using different definitions of objects and attributes on different levels.

The above two issues make the FCA based information retrieval from web usage data a challenging problem. For future work, we can implement a FCA based information retrieval system, which should help users to browse web usage data, provide potential search terms, and generalize/specialize search results from multi-level and multi-dimensional viewpoints.

8.2.5 Extending the Semantic Web Personalization Approach

In this research, we have applied our proposed approaches to support effective personalized services on the Semantic Web. However, several issues still have to be resolved, which are listed as follows.

Automatic Web Usage Ontology Integration

In this research, we have proposed an automatic approach for generating Web Usage Ontologies of registered users of a website. In practice, it will be very costly if all websites only build and maintain their own Web Usage Ontologies. Since all Web Usage Ontologies are sharable on the Semantic Web, a website may import Web Usage Ontologies from other websites with resources in similar domains or merge it into its own Web Usage Ontologies. Ontology integration [CGL01] is a process for merging existing ontologies or defining mapping rules between these ontologies in order to allow importing and reusing of existing ontologies.

Formal Concept Analysis is not only an effective technique for automatic ontology generation, but also a potential technique for automatic ontology integration. In [SM01], Stumme *et al.* proposed FCA-MERGE, a bottom-up technique for merging ontologies semi-automatically. However, this approach still requires human experts to generate the integration rules. For future work, we can investigate novel FCA based techniques for automatic integration of Web Usage Ontologies.

Incremental Updating of Web Usage Ontology

In practice, web usage logs of a website are constantly updated all the time. The access behaviors of users also change from day to day. In this research, we assume the users' interests are relatively static for a short period of time and only require to update the Web Usage Ontology when necessary. However, the updating process of Web Usage Ontology is very costly. Therefore, it is necessary to develop an incremental method for updating Web Usage Ontology. In order to maintain Web Usage Ontology incrementally, we should consider the following issues:

- *Reuse the existing Web Usage Ontology.* To make use of the existing user behavior model, the incremental updating should be performed during the time when users' interests have not changed much. Therefore, an efficient mechanism is required to detect and measure the changes of users' interests.

- *Maintain the deletions of invalid resources and the insertions of new resources.* Sometimes, a part of resources in the domain ontology of the website may not be valid anymore. Therefore, we need to discard such resources from the current Web Usage Ontology. Conversely, new resources are required to be added into the existing Web Usage Ontology when needed.

8.2.6 Supporting Other Semantic Web Personalization Services

In this research, we have focused on investigating novel web usage mining techniques to support efficient and effective personalized web services on the Web and Semantic Web. For future work, we can further extend the proposed techniques to support other Semantic Web personalization services such as e-learning [SSS01, KG03] on the Semantic Web.

Recently, the rapid development of information technology has facilitated the use of e-learning techniques to provide online personalized learning services for users, such as Intelligent Tutoring Systems (ITS) [RNS05]. With the development of the Semantic Web services, e-learning services can be delivered on the Semantic Web. To achieve this, this research can be extended to support personalized services for Semantic Web e-learning that not only can assist registered learners to learn and enhance their knowledge on specific domains, but also can help teachers and schools from different locations share and manage their knowledge, learning materials and teaching experience online.

Firstly, the Semantic Web techniques can be incorporated into the current e-learning framework for representing semi-structured data, such as domain knowledge, learning materials and user profiles, in order to promote interoperability and standardization for Semantic Web e-learning. As such, the domain knowledge and learning materials can be imported, managed, shared and reused in the form of ontologies on the Semantic Web. The learners and teachers can then browse and search the knowledge and learning materials they want by means of conceptual navigation and semantic querying.

In addition, Semantic Web personalization techniques can be applied for effective personalized learning services. Information on user interactions with the system (such as the answers to questions, the topics of learning materials they searched, etc.) are monitored and recorded into the interaction logs, which is similar to the traditional web usage logs, but with more semantic information for the e-learning purpose. Based on the interaction logs, users' interests, intentions, preferences and performance can be discovered automatically using the techniques proposed in this research, and then used for constructing the personal profiles of

individual users for supporting personalized learning services to users, such as recommending related learning materials to the users, personal learning strategies, etc.

Appendix A

List of Publications

The work reported in this thesis has been accepted/published in the following international journals, conferences and workshops.

A.1 International Journal Papers

1. Baoyao Zhou, Siu Cheung Hui, and Kuiyu Chang. Enhancing Mobile Web Access using Intelligent Recommendations. *IEEE Intelligent Systems*, 21(1):28-34, 2006.
2. Baoyao Zhou, Siu Cheung Hui, and Alvis C. M. Fong. Efficient Sequential Access Pattern Mining for Web Recommendations. To be published in *International Journal of Knowledge-Based and Intelligent Engineering Systems (KES Journal)*, 2006.
3. Baoyao Zhou, Siu Cheung Hui, and Alvis C. M. Fong. A Web Usage Lattice Based Mining Approach for Intelligent Web Personalization. *International Journal of Web Information Systems (IJWIS)*, 1(3):137-145, 2005.

A.2 International Conference / Workshop Papers

1. Baoyao Zhou, Siu Cheung Hui, and Alvis C. M. Fong. An Effective Approach for Periodic Web Personalization. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 284-292, Hong Kong, China, December 18-22, 2006.
2. Baoyao Zhou, Siu Cheung Hui, and Alvis C. M. Fong. Discovering and Visualizing Temporal-based Web Access Behavior. In *WI '05: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 297-300, Compiegne University of Technology, France, September 19-22, 2005.

3. Baoyao Zhou, Siu Cheung Hui, and Alvis C. M. Fong. Web Usage Mining for Semantic Web Personalization. In *PerSWeb '05: Proceedings of the Workshop on Personalization on the Semantic Web*, pages 66-72, in conjunction with the 10th International Conference on User Modeling (UM'05), Edinburgh, UK, July 24-25, 2005.
4. Baoyao Zhou, Siu Cheung Hui, and Kuiyu Chang. An Intelligent Recommender System using Sequential Web Access Patterns. In *CIS '04: Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems*, Vol. 1, pages 393-398, Singapore, December 1-3, 2004.
5. Baoyao Zhou, Siu Cheung Hui, and Kuiyu Chang. A Formal Concept Analysis Approach for Web Usage Mining. In *ICIP: 04: Proceedings of the 2004 International Conference on Intelligent Information Processing*, pages 437-441, Beijing, China, October 21-23, 2004.
6. Baoyao Zhou, Siu Cheung Hui, and Alvis C. M. Fong. An Efficient Approach for Mining Periodic Sequential Access Patterns. In *PRICAI '04: Proceedings of the 8th Pacific Rim International Conference on Artificial Intelligence, PRICAI 2004: Trends in Artificial Intelligence, Auckland, New Zealand, August 9-13, 2004, LNAI*, Vol. 3157, pages 485-494. Springer, 2004.
7. Baoyao Zhou, Siu Cheung Hui, and Alvis C. M. Fong. CS-mine: An Efficient WAP-tree Mining for Web Access Patterns. In *APWEB '04: Proceedings of the 6th Asia Pacific Web Conference, Hangzhou, China, April 14-17, 2004, Advanced Web Technologies and Applications, LNCS*, Vol. 3007, pages 523-532. Springer, 2004.

Bibliography

- [AAP01] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. A Tree Projection Algorithm for Generation of Frequent Item Sets. *Journal of Parallel and Distributed Computing*, 61(3):350–371, 2001.
- [ABA04] M. Agyemang, K. Barker, and R. Alhajj. Framework for Mining Web Content Outliers. In *SAC '04: Proceedings of the 2004 ACM Symposium on Applied Computing*, pages 590–594, Nicosia, Cyprus, March 2004. ACM Press.
- [Abr03] A. Abraham. Business Intelligence from Web Usage Mining. *Journal of Information & Knowledge Management (JIKM)*, 2(4):375–390, 2003.
- [AEMT00] K.I M. Ahmed, N. M. El-Makky, and Y. Taha. A Note on “Beyond Market Baskets: Generalizing Association Rules to Correlations”. *SIGKDD Explor. Newsl.*, 1(2):46–48, 2000.
- [AS94] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, Santiago de Chile, Chile, Sept. 1994. Morgan Kaufmann.
- [AS95] R. Agrawal and R. Srikant. Mining Sequential Patterns. In *ICDE '95: Proceedings of the 11th International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, March 1995. IEEE Computer Society Press.
- [BEH⁺02] E. Bozsak, M. Ehrig, S. Handschuh, A. Hotho, A. Maedche, B. Motik, D. Oberle, C. Schmitz, S. Staab, L. Stojanovic, N. Stojanovic, R. Studer, G. Stumme, Y. Sure, J. Tane, R. Volz, and V. Zacharias. KAON - Towards A Large Scale Semantic Web. In *EC-WEB '02: Proceedings of the 3rd International Conference on E-Commerce and Web Technologies*, pages 304–313, Aix-en-Provence, France, Sept. 2002. Springer-Verlag.

- [Ber01] M. K. Bergman. The Deep Web: Surfacing Hidden Value. *Journal of Electronic Publishing*, 7(1), 2001.
- [BFR⁺03] M. Baglioni, U. Ferrara, A. Romei, S. Ruggieri, and F. Turini. Preprocessing and Mining Web Log Data for Web Personalization. In *AI*IA 03: Advances in Artificial Intelligence, Proceedings of the 8th Congress of the Italian Association for Artificial Intelligence*, pages 237–249, Pisa, Italy, Sept. 2003. Springer-Verlag.
- [BG00] A. Banerjee and J. Ghosh. Concept-based Clustering of Clickstream Data. In *ICIT '00: Proceedings of the 3rd International Conference on Information Technology*, pages 145–150, Dec. 2000.
- [BHM⁺04] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, and C. Ferris D. Orchard. Web Services Architecture. W3C Working Group Note, Feb. 2004. Available at <http://www.w3.org/TR/ws-arch/>.
- [BHS02] B. Berendt, A. Hotho, and G. Stumme. Towards Semantic Web Mining. In *ISWC '02: Proceedings of the 1st International Semantic Web Conference on the Semantic Web*, pages 264–278, Sardinia, Italy, June 2002. Springer-Verlag.
- [BL92] T. Berners-Lee. Hypertext Markup Language (HTML), 1992. Available at <http://www.w3.org/MarkUp/>.
- [BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001.
- [BMS97] S. Brin, R. Motwani, and C. Silverstein. Beyond Market Baskets: Generalizing Association Rules to Correlations. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 265–276, New York, NY, USA, 1997. ACM Press.
- [BMUT97] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic Itemset Counting and Implication Rules for Market Basket Data. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 255–264, New York, NY, USA, 1997. ACM Press.
- [BV03] S. Bansal and J. M. Vidal. Matchmaking of Web Services based on the DAML-S Service Model. In *AAMAS '03: Proceedings of the 2nd International Joint*

- Conference on Autonomous Agents & Multiagent Systems*, pages 926–927, Melbourne, Australia, July 2003. ACM Press.
- [CCH01] P. Clerkin, P. Cunningham, and C. Hayes. Ontology Discovery for the Semantic Web Using Hierarchical Clustering. In *Proceedings of the 1st Semantic Web Mining Workshop, at ECML/PKDD '01*, 2001.
- [CCMW01] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. W3C Note, March 2001. Available at <http://www.w3.org/TR/wsdl>.
- [CGL01] D. Calvanese, G. De Giacomo, and M. Lenzerini. A Framework for Ontology Integration. In *SWWS '01: Proceedings of the 1st Semantic Web Working Symposium*, pages 303–316, Stanford University, California, USA, July 2001.
- [Cha02] S. Chakrabarti. *Mining the Web: Discovering Knowledge from HyperText Data*. Morgan Kaufmann, Oct. 2002.
- [CKK02] Y. H. Cho, J. K. Kim, and S. H. Kim. A Personalized Recommender System Based on Web Usage Mining and Decision Tree Induction. *Expert Systems with Applications*, 23(3):329–342, 2002.
- [CMS99] R. Cooley, B. Mobasher, and J. Srivastava. Data Preparation for Mining World Wide Web Browsing Patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.
- [CPY96] M. S. Chen, J. S. Park, and P. S. Yu. Data Mining for Path Traversal Patterns in a Web Environment. In *ICDCS '96: Proceedings of the 16th International Conference on Distributed Computing Systems*, pages 385–392. IEEE Computer Society, May 1996.
- [CR96] C. Carpineto and G. Romano. Information Retrieval through Hybrid Navigation of Lattice Representations. *International Journal of Human Computer Studies*, 45(5):553–578, 1996.
- [CR05] C. Carpineto and G. Romano. Using Concept Lattices for Text Retrieval and Mining. In *Formal Concept Analysis, Foundations and Applications*, volume 3626 of *Lecture Notes in Computer Science*, pages 161–179. Springer-Verlag, 2005.

- [CST03] P. Cimiano, S. Staab, and J. Tane. Deriving Concept Hierarchies from Text by Smooth Formal Concept Analysis. In *LLWA '03: Proceedings of GI Workshop Lehren-Lernen-Wissen-Adaptivtit*, pages 72–79, Karlsruhe, Germany, 2003.
- [DM02] H. Dai and B. Mobasher. Using Ontologies to Discover Domain-level Web Usage Profiles. In *Proceedings of the 2nd Semantic Web Mining Workshop at ECML/PKDD*, 2002.
- [EL05] C. I. Ezeife and Y. Lu. Mining Web Log Sequential Patterns with Position Coded Pre-Order Linked WAP-Tree. *Data Mining and Knowledge Discovery*, 10(1):5–38, 2005.
- [EV03] M. Eirinaki and M. Vazirgiannis. Web Mining for Web Personalization. *ACM Transactions on Internet Technology (TOIT)*, 3(1):1–27, 2003.
- [EVV03] M. Eirinaki, M. Vazirgiannis, and I. Varlamis. SEWeP: Using Site Semantics and a Taxonomy to Enhance the Web Personalization Process. In *KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108, Washington, D.C., USA, 2003. ACM Press.
- [FBH00] X. Fu, J. Budzik, and K. J. Hammond. Mining Navigation History for Recommendation. In *IUI '00: Proceedings of the 5th international conference on Intelligent user interfaces*, pages 106–112, New York, NY, USA, 2000. ACM Press.
- [FK97] H. Frigui and R. Krishnapuram. Clustering by Competitive Agglomeration. *Pattern Recognition*, 30(7):1109–1119, 1997.
- [FL05] F. M. Facca and P. L. Lanzi. Mining Interesting Knowledge from Weblogs: A Survey. *Data & Knowledge Engineering*, 53(3):225–241, 2005.
- [FLM98] D. Florescu, A. Levy, and A. Mendelzon. Database Techniques for the World-Wide Web: A Survey. *ACM SIGMOD Record*, 27(3):59–74, 1998.
- [FMM03] P. Fraternali, M. Matera, and A. Maurino. Conceptual-Level Log Analysis for the Evaluation of Web Application Quality. In *LA-WEB '03: Proceedings of the 1st Latin American Web Congress, Empowering Our Web*, pages 46–57, Sanitago, Chile, Nov. 2003. IEEE Computer Society.

- [FS04] X. Fang and O. R. L. Sheng. LinkSelector: A Web Mining Approach to Hyperlink Selection for Web Portals. *ACM Transactions on Internet Technology (TOIT)*, 4(2):209–237, 2004.
- [GH03] M. Gery and H. Haddad. Evaluation of Web Usage Mining Approaches for User’s Next Request Prediction. In *WIDM ’03: Proceedings of the 5th ACM International Workshop on Web Information and Data Management*, pages 74–81, New Orleans, Louisiana, USA, 2003. ACM Press.
- [GMA93] R. Godin, R. Missaoui, and A. April. Experimental Comparison of Navigation in a Galois Lattice with Conventional Information Retrieval Methods. *International Journal of Man-Machine Studies*, 38(5):747–767, 1993.
- [GMA95] R. Godin, R. Missaoui, and H. Alaoui. Incremental Concept Formation Algorithms based on Galois (Concept) Lattices. *Computational Intelligence*, 11:246–267, 1995.
- [Goo] Google Personalized Web Search. Available at <http://labs.google.com/personalized>.
- [GSW04] B. Ganter, G. Stumme, and R. Wille. Formal Concept Analysis: Theory and Applications. *The Journal of Universal Computer Science*, 10(8):926, 2004.
- [GW97] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1997.
- [HB99] S. Hettich and S. D. Bay. The UCI KDD Archive [<http://kdd.ics.uci.edu>], 1999.
- [HC02] J. Han and K. Chang. Data Mining for Web Intelligence. *IEEE Computer*, 35(11):64–70, 2002.
- [HC04] Kuo-Yu Huang and Chia-Hui Chang. Asynchronous Periodic Patterns Mining in Temporal Databases. In *Proceedings of IASTED International Conference on Databases and Applications, part of the 22nd Multi-Conference on Applied Informatics*, pages 43–48, Innsbruck, Austria, 2004.
- [HDY99] J. Han, G. Dong, and Y. Yin. Efficient Mining of Partial Periodic Patterns in Time Series Database. In *ICDE ’99: Proceedings of the 15th International Conference on Data Engineering*, pages 106–115, Sydney, Australia, March 1999. IEEE Computer Society Press.

- [Her04] E. Herder. Sniffing Around for Providing Navigation Assistance. In *Proceedings of Workshop on Adaptivity and User Modeling in Interactive Systems ABIS '04*, pages 20–24, Berlin, Germany, 2004.
- [HK05] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
- [HKKM98] E.-H. Han, G. Karypis, V. Kumar, and B. Mobasher. Hypergraph Based Clustering in High-Dimensional Data Sets: A Summary of Results. *IEEE Data(base) Engineering Bulletin*, 21(1):15–22, 1998.
- [HPAS04] X.i Huang, F. Peng, A. An, and D. Schuurmans. Dynamic Web Log Session Identification with Statistical Language Models. *Journal of the American Society for Information Science and Technology (JASIST)*, 55(14):1290–1303, 2004.
- [HPMA⁺00] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. FreeSpan: Frequent Pattern-projected Sequential Pattern Mining. In *KDD '00: Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 355–359, Boston, Massachusetts, USA, 2000. ACM Press.
- [HPY00] J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 1–12, Dallas, Texas, USA, 2000. ACM Press.
- [HSWW00] J. Hereth, G. Stumme, R. Wille, and U. Wille. Conceptual Knowledge Discovery and Data Analysis. In *ICCS '00: Proceedings of the 8th International Conference on Conceptual Structures, Logical, Linguistic, and Computational Issues*, pages 421–437, Darmstadt, Germany, Aug. 2000. Springer-Verlag.
- [JdR05] V. Jijkoun and M. d. Rijke. Retrieving Answers from Frequently Asked Questions Pages on the Web. In *CIKM '05: Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 76–83, Bremen, Germany, 2005. ACM Press.
- [JFM97] T. Joachims, D. Freitag, and T. M. Mitchell. Web Watcher: A Tour Guide for the World Wide Web. In *IJCAI '97: Proceedings of the 15th International*

- Joint Conference on Artificial Intelligence*, pages 770–777, Nagoya, Japan, Aug. 1997.
- [JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Computing Surveys*, 31(3):264–323, 1999.
- [Kaz04] P. Kazienko. Multi-agent Web Recommendation Method Based on Indirect Association Rules. In *KES '04: Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information and Engineering Systems*, Wellington, New Zealand, Spet. 2004.
- [KB00] R. Kosala and H. Blockeel. Web Mining Research: A Survey. *ACM SIGKDD Explorations*, 2(1):1–15, 2000.
- [KC01] M. Kim and P. Compton. Formal Concept Analysis for Domain-Specific Document Retrieval systems. In *AI '01: Advances in Artificial Intelligence, Proceedings of the 14th Australian Joint Conference on Artificial Intelligence*, pages 237–248, Adelaide, Australia, Dec. 2001.
- [KC04] G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and abstract syntax. W3C Recommendation, Feb. 2004. Available at <http://www.w3.org/TR/rdf-concepts/>.
- [KG03] V. Kolovski and J. Galletly. Towards E-learning via the Semantic Web. In *CompSysTech '03: Proceedings of the 4th International Conference on Computer Systems and Technologies*, pages 591–596, Rousse, Bulgaria, 2003. ACM Press.
- [KJ04] P. Kolari and A. Joshi. Web Mining: Research and Practice. *Computing in Science and Engineering*, 6(4):49–53, 2004.
- [KJ05] T. Kamdar and A. Joshi. Using Incremental Web Log Mining to Create Adaptive Web Servers. *International Journal on Digital Libraries*, 5(2):133–150, 2005.
- [KMM⁺97] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM (CACM)*, 40(3):77–87, 1997.

- [Knu98] D. E. Knuth. *The Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley Publishing Company, 2nd edition, 1998.
- [Koh88] T. Kohonen. Self-Organized Formation of Topologically Correct Feature Maps. *Neurocomputing: foundations of research*, pages 509–521, 1988.
- [KY95] G.J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall, Upper Saddle River, NJ, 1995.
- [LAR02] W. Lin, S. A. Alvarez, and C. Ruiz. Efficient Adaptive-Support Association Rule Mining for Recommender Systems. *Data Mining and Knowledge Discovery*, 6(1):83–105, 2002.
- [LCN03] B. Liu, C. W. Chin, and H. T. Ng. Mining Topic-Specific Concepts and Definitions on the Web. In *WWW '03: Proceedings of the 12th International Conference on World Wide Web*, pages 251–260, Budapest, Hungary, 2003. ACM Press.
- [LD05] D. Li and J. S. Deogun. Discovering Partial Periodic Sequential Association Rules with Time Lag in Multiple Sequences for Prediction. In *ISMIS '05: Proceedings of the 15th International Symposium on Foundations of Intelligent Systems*, pages 332–341, Saratoga Springs, NY, USA, 2005.
- [LHF02] P. Y. Lee, S. C. Hui, and A. C. M. Fong. Neural Networks for Web Content Filtering. *IEEE Intelligent Systems*, 17(5):48–57, 2002.
- [Lie95] H. Lieberman. Letizia: An Agent that Assists Web Browsing. In *IJCAI '95: Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 924–929, Montreal, Quebec, Canada, 1995. Morgan Kaufmann.
- [LK05] C.-H. Li and C.-C. Kit. Web Structure Mining for Usability Analysis. In *WI '05: Proceedings of the The 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 309–312, Compiegne, France, Sept. 2005. IEEE Computer Society.
- [LLLY02] W. Lou, G. Liu, H. Lu, and Q. Yang. Cut-and-Pick Transactions for Proxy Log Mining. In *EDBT '02: Proceedings of the 8th International Conference on Extending Database Technology*, pages 88–105, London, UK, 2002. Springer-Verlag.

- [LLS03] H. Lu, Q. Luo, and Y. K. Shun. Extending a Web Browser with Client-Side Mining. In *APWeb '03: Proceedings of the 5th Asian-Pacific Web Conference, Web Technologies and Applications*, pages 166–177, Xian, China, 2003.
- [LNWJ01] Y. Li, P. Ning, X. S. Wang, and S. Jajodia. Discovering Calendar-Based Temporal Association Rules. In *TIME '01: Proceedings of the 8th International Symposium on Temporal Representation and Reasoning*, pages 111–118, Cividale del Friuli, Italy, June 2001. IEEE Computer Society.
- [LNWJ03] Y. Li, P. Ning, X. S. Wang, and S. Jajodia. Discovering Calendar-Based Temporal Association Rules. *Data & Knowledge Engineering*, 44(2):193–218, 2003.
- [MBH⁺04] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. OWL-S: Semantic Markup for Web Services. W3C Member Submission, Nov. 2004. Available at <http://www.w3.org/Submission/OWL-S/>.
- [MCE04] E.-S. Maged, R. Carolina, and A. R. Elke. FS-Miner: Efficient and Incremental Mining of Frequent Sequence Patterns in Web Logs. In *WIDM '04: Proceedings of the 6th Annual ACM International Workshop on Web Information and Data Management*, pages 128–135, Washington DC, USA, 2004. ACM Press.
- [MCS99] B. Mobasher, R. Cooley, and J. Srivastava. Creating Adaptive Web Sites Through Usage-Based Clustering of URLs. In *KDEX '99: Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange*, page 19. IEEE Computer Society, 1999.
- [MDLN01] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective Personalization Based on Association Rule Discovery from Web Usage Data. In *WIDM '01: Proceedings of the 3rd International Workshop on Web Information and Data Management , at ACM CIKM '01*, pages 9–15, Atlanta, Georgia, USA, 2001.
- [MDLN02] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Using Sequential and Non-Sequential Patterns in Predictive Web Usage Mining Tasks. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 669–672, Maebashi City, Japan, Dec. 2002. IEEE Computer Society.
- [MGPL04] M. N. Moreno, F. J. García, M. J. Polo, and V. F. López. Using Association Analysis of Web Data in Recommender Systems. In *EC-Web '04: Proceedings*

of the 5th International Conference on E-Commerce and Web Technologies, volume 3182 of *Lecture Notes in Computer Science*, pages 11–20, Zaragoza, Spain, Aug. 2004. Springer.

- [MH04] D. L. McGuinness and F. Van Harmelen. OWL Web Ontology Language Overview. W3C Recommendation, Feb. 2004. Available at <http://www.w3.org/TR/owl-features/>.
- [Mit03] N. Mitra. SOAP Version 1.2 Part 0: Primer. W3C Recommendation, June 2003. Available at <http://www.w3.org/TR/soap12-part0/>.
- [MKR04] B. N. Miller, J. A. Konstan, and J. Riedl. PocketLens: Toward a Personal Recommender System. *ACM Transactions on Information Systems (TOIS)*, 22(3):437–476, 2004.
- [MLME04] R. Meo, P. L. Lanzi, M. Matera, and R. Esposito. Integrating Web Conceptual Modeling and Web Usage Mining. In *Proceedings of the WebKDD Workshop on Web Mining and Web Usage Analysis*, pages 105–115, 2004.
- [Mob99] B. Mobasher. A Web Personalization Engine Based on User Transaction Clustering. In *WITS '99: Proceedings of the 9th Workshop on Information Technologies and Systems*, 1999.
- [MS00a] A. Maedche and S. Staab. Discovering Conceptual Relations from Text. In *ECAI '00: Proceedings of the 14th European Conference on Artificial Intelligence*, pages 321–325, Berlin, Germany, Aug. 2000. IOS Press.
- [MS00b] B. M. Masand and M. Spiliopoulou, editors. *Web Usage Analysis and User Profiling*, volume 1836 of *Lecture Notes in Computer Science*, San Diego, California, USA, Aug. 2000. Springer.
- [MS01] A. Maedche and S. Staab. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.
- [NFJK99] O. Nasraoui, H. Frigui, A. Joshi, and R. Krishnapuram. Mining Web Access Logs Using Relational Competitive Fuzzy Clustering. In *IFSA '99: Proceedings of the 8th International Fuzzy Systems Association World Congress*, Taipei, Taiwan, Aug. 1999.

- [NM01] N. F. Noy and D. L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. Technical report, Stanford Medical Informatics, Palo Alto, 2001.
- [NR99] L. Nourine and O. Raynaud. A Fast Algorithm for Building Lattices. *Information Processing Letters*, 71(5-6):199–204, 1999.
- [NSER02] N. Niu, E. Stroulia, and M. El-Ramly. Understanding Web Usage for Dynamic Web-Site Adaptation: A Case Study. In *WSE '02: Proceedings of the 4th International Workshop on Web Site Evolution*, page 53, Montreal, Canada, Oct. 2002. IEEE Computer Society.
- [OAS04] OASIS. Universal Description, Discovery, and Integration (UDDI) protocol, 2004. Available at <http://www.uddi.org>.
- [OBHG03] D. Oberle, B. Berendt, A. Hotho, and J. Gonzalez. Conceptual User Tracking. In *AWIC '03: Proceedings of the 1st International Atlantic Web Intelligence Conference*, volume 2663 of *Lecture Notes in Computer Science*, pages 155–164, Madrid, Spain, May 2003. Springer-Verlag.
- [ORS98] B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic Association Rules. In *ICDE '98: Proceedings of the 14th International Conference on Data Engineering*, pages 412–421, Orlando, Florida, USA, 1998. IEEE Computer Society.
- [OS03] D. Olaru and B. Smith. Modelling Daily Activity Schedules with Fuzzy Logic. In *Proceedings of the 10th International Conference on Travel Behaviour Research*, Lucerne, Switzerland, 2003. Elsevier.
- [PE00] M. Perkowitz and O. Etzioni. Adaptive Web Sites. *Communications of the ACM*, 43(8):152–158, 2000.
- [PHMA⁺01] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In *ICDE '01: Proceedings of the 17th International Conference on Data Engineering*, pages 215–224, Heidelberg, Germany, April 2001. IEEE Computer Society.
- [PHMAZ00] J. Pei, J. Han, B. Mortazavi-Asl, and H. Zhu. Mining Access Patterns Efficiently from Web Logs. In *PAKDD '00: Proceedings of the 4th Pacific-Asia*

- Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, pages 396–407, Kyoto, Japan, 2000. Springer-Verlag.
- [PHWZ05] D. Peng, S. Huang, X. Wang, and A. Zhou. Concept-Based Retrieval of Alternate Web Services. In *DASFAA '05: Proceedings of the 10th International Conference on Database Systems for Advanced Applications*, April 2005.
- [PP99] P. L. T. Pirolli and J. E. Pitkow. Distributions of Surfers' Paths through the World Wide Web: Empirical Characterizations. *World Wide Web*, 2(1-2):29–45, 1999.
- [PPPS03] D. Pierrakos, G. Paliouras, C. Papatheodorou, and C. D. Spyropoulos. Web Usage Mining as a Tool for Personalization: A Survey. *User Modeling and User-Adapted Interaction*, 13(4):311–372, 2003.
- [PSR03] N. Prakash, P. Selva, and Ramachandran. Personalized Surfing via Web Mining, 2003. Available at <http://cs.engr.uky.edu/dekhtyar/685-Spring2003/project/Group8.pdf>.
- [QHFC04] T. T. Quan, S. C. Hui, A. C. M. Fong, and T. H. Cao. Automatic Generation of Ontology for Scholarly Semantic Web. In *ISWC '04: Proceedings of the 3rd International Semantic Web Conference*, pages 726–740, Hiroshima, Japan, Nov. 2004.
- [Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
- [RMS98] S. Ramaswamy, S. Mahajan, and A. Silberschatz. On the Discovery of Interesting Patterns in Association Rules. In *VLDB '98: Proceedings of the 24th International Conference on Very Large Data Bases*, pages 368–379, New York, USA, Aug. 1998. Morgan Kaufmann Publishers Inc.
- [RNS05] M. Rodrigues, P. Novais, and M. F. Santos. Future Challenges in Intelligent Tutoring Systems c A Framework. In *Recent Research Developments in Learning Technologies*, pages 129–134, Badajoz, Spain, 2005. FORMATEX.
- [RPG02] C. G. Rodriguez and P. Palao-Gostanza. Ergotracer: An Internet User Behaviour Tracer. In *IICS '02: Proceedings of the 2nd International Workshop on*

- Innovative Internet Computing Systems*, pages 121–132, Kühlungsborn, Germany, 2002. Springer-Verlag.
- [SA96] R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *EDBT '96: Proceedings of the 5th International Conference on Extending Database Technology*, pages 3–17, Avignon, France, 1996. Springer-Verlag.
- [SBH02] G. Stumme, B. Berendt, and A. Hotho. Usage Mining for and on the Semantic Web. In *Next Generation Data Mining, Proceedings of NSF Workshop*, pages 77–86, Baltimore, Nov. 2002.
- [SCDT00] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. *ACM SIGKDD Explorations*, 1(2):12–23, 2000.
- [SEA⁺02] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative Ontology Development for the Semantic Web. In *ISWC '02: Proceedings of the First International Semantic Web Conference on the Semantic Web*, pages 221–235, Sardinia, Italy, June 2002. Springer-Verlag.
- [SKCM01] C. Shahabi, F. B. Kashani, Y.-S. Chen, and D. McLeod. Yoda: An Accurate and Scalable Web-Based Recommendation System. In *CoopIS '01: Proceedings of the 9th International Conference on Cooperative Information Systems*, volume 2172 of *Lecture Notes in Computer Science*, pages 418–432, Trento, Italy, Sept. 2001. Springer.
- [SM01] G. Stumme and A. Maedche. FCA-MERGE: Bottom-Up Merging of Ontologies. In *IJCAI '01: Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 225–234, Seattle, Washington, USA, Aug. 2001. Morgan Kaufmann.
- [SOBB03] D. M. Sow, D. P. Olsheski, M. Beigi, and G. Banavar. Prefetching Based on Web Usage Mining. In *Middleware '03: Proceedings of ACM/IFIP/USENIX International Middleware Conference*, pages 262–281, Rio de Janeiro, Brazil, June 2003.
- [SP04] S. C. K. Shiu and S. K. Pal. Case-Based Reasoning: Concepts, Features and Soft Computing. *Applied Intelligence*, 21(3):233–238, 2004.

- [Spi00] M. Spiliopoulou. Web Usage Mining for Web Site Evaluation. *Communications of the ACM (CACM)*, 43(8):127–134, 2000.
- [SSS01] L. Stojanovic, S. Staab, and R. Studer. eLearning based on the Semantic Web. In *WebNet '01: Proceedings of the 2001 World Conference on the WWW and Internet*, Orlando, Florida, USA, Oct. 2001.
- [STB⁺01] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Intelligent Structuring and Reducing of Association Rules with Formal Concept Analysis. In *KI '01: Proceedings of the Joint German/Austrian Conference on AI, Advances in Artificial Intelligence*, pages 335–350, Vienna, Austria, Sept. 2001. Springer-Verlag.
- [STB⁺02] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Computing Iceberg Concept Lattices with TITANIC. *Data Knowledge Engineering.*, 42(2):189–222, 2002.
- [STBL01] G. Stumme, R. Taouil, Y. Bastide, and L. Lakhal. Conceptual Clustering with Iceberg Concept Lattices. In *Proceedings of GI-Fachgruppentreffen Maschinelles Lernen '01*, Oct. 2001.
- [Stu02] G. Stumme. Efficient Data Mining based on Formal Concept Analysis. In *DEXA '02: Proceedings of the 13th International Conference on Database and Expert Systems Applications*, pages 534–546, Aix-en-Provence, France, Sept. 2002.
- [Sur] Surfaid Analytics. Available at <http://surfaid.dfw.ibm.com>.
- [SWW98] G. Stumme, R. Wille, and U. Wille. Conceptual Knowledge Discovery in Databases Using Formal Concept Analysis Methods. In *PKDD '98: Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 450–458, Nantes, France, Sept 1998. Springer-Verlag.
- [Tan99] A.-H. Tan. Text Mining: The State of the Art and the Challenges. In *Proceedings of Workshop on Knowledge Discovery from Advanced Databases, at PAKDD '99*, pages 65–70, 1999.
- [TBGR00] A. Todirascu, F. De Beuvron, Dan Gâlea, and F. Rousselot. Using Description Logics for Ontology Extraction. In *OL '00: Proceedings of the 1st Workshop on Ontology Learning at the 14th European Conference on Artificial Intelligence*

- ECAI '00*, volume 31 of *CEUR Workshop Proceedings*, Berlin, Germany, Aug. 2000.
- [TK02] P.-N. Tan and V. Kumar. Discovery of Web Robot Sessions Based on Their Navigational Patterns. *Data Mining & Knowledge Discovery.*, 6(1):9–35, 2002.
- [VMG04] P. Valtchev, R. Missaoui, and R. Godin. Formal Concept Analysis for Knowledge Discovery and Data Mining: The New Challenges. In *ICFCA '04: Proceedings of the 2nd International Conference on Formal Concept Analysis*, pages 352–371, Sydney, Australia, 2004.
- [VW95] F. Vogt and R. Wille. TOSCANA - A Graphical Tool for Analyzing and Exploring Data. In *GD '94: Proceedings of the DIMACS International Workshop on Graph Drawing*, pages 226–233, Princeton, New Jersey, USA, Oct. 1995. Springer-Verlag.
- [WAS05] X. Wang, A. Abraham, and K. A. Smith. Intelligent Web Traffic Mining and Analysis. *Journal of Network and Computer Applications*, 28(2):147–165, 2005.
- [Web] WebTrends Web Analytics. Available at <http://www.netiq.com/webtrends>.
- [WSP01] C. Wong, S. Shiu, and S. Pal. Mining Fuzzy Association Rules for Web Access Case Adaptation. In *Proceedings of the 4th International Conference on Case-Based Reasoning, Case-Based Reasoning Research and Development*, 2001.
- [XP01] Y. Xie and V. V. Phoha. Web User Clustering from Access Log using Belief Function. In *K-CAP '01: Proceedings of the 1st International Conference on Knowledge Capture*, pages 202–208, Victoria, BC, Canada, Oct. 2001. ACM Press.
- [YBP⁺04] F. Yergeau, T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler. Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation, Feb. 2004. Available at <http://www.w3.org/TR/2004/REC-xml-20040204/>.
- [YP02] H. Yang and S. Parthasarathy. On the Use of Constrained Associations for Web Log Mining. In *WEBKDD '02: Proceedings of the 4th International Workshop on Mining Web Data for Discovering Usage Patterns and Profiles*, volume 2703 of *Lecture Notes in Computer Science*, pages 100–118, Edmonton, Canada, July 2002. Springer.

- [YWY00] J. Yang, W. Wang, and P. S. Yu. Mining Asynchronous Periodic Patterns in Time Series Data. In *KDD '00: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 275–279, Boston, Massachusetts, USA, 2000. ACM Press.
- [YZL01] Q. Yang, H. H. Zhang, and T. Li. Mining Web Logs for Prediction Models in WWW Caching and Prefetching. In *KDD '01: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 473–478, San Francisco, California, 2001. ACM Press.
- [Zad65] L. A. Zadeh. Fuzzy Sets. *Journal of Information and Control*, 8:338–353, 1965.
- [Zad75] L. A. Zadeh. Fuzzy Logic and Approximate Reasoning. *Synthese*, 30:407–428, 1975.
- [ZC02] F. Zhang and H. Chang. Research and Development in Web Usage Mining Systems: Key Issues and Proposed Solutions: A Survey. In *ICMLC '02: Proceedings of the 1st IEEE International Conference on Machine Learning and Cybernetics*, pages 986–C990. IEEE Computer Society, Nov. 2002.
- [ZCS⁺01] B. Zhou, J. Chen, J. Shi, H. Zhang, and Q. Wu. Website Link Structure Evaluation and Improvement Based on User Visiting Patterns. In *HYPertext '01: Proceedings of the 12th ACM Conference on Hypertext and Hypermedia*, pages 241–244, Århus, Denmark, 2001. ACM Press.
- [ZHC04] B. Zhou, S. C. Hui, and K. Chang. An Intelligent Recommender System using Sequential Web Access Patterns. In *CIS '04: Proceedings of 2004 IEEE Conference on Cybernetics and Intelligent Systems*, pages 393–398, Singapore, 2004.
- [ZHF04] B. Zhou, S. C. Hui, and A. C. M. Fong. CS-Mine: An Efficient WAP-Tree Mining for Web Access Patterns. In *APWeb '04: Proceedings of the 6th Asia-Pacific Web Conference, Advanced Web Technologies and Applications*, volume 3007 of *Lecture Notes in Computer Science*, pages 523–532, Hangzhou, China, 2004. Springer.
- [ZHF05] B. Zhou, S. C. Hui, and A. C. M. Fong. Discovering and Visualizing Temporal-Based Web Access Behavior. In *WI '05: Proceedings of 2005 IEEE / WIC /*

ACM International Conference on Web Intelligence, pages 297–300, Compiègne, France, 2005.

- [ZHL⁺98] O. R. Zaiane, J. Han, Z.-N. Li, S. H. Chee, and J. Y. Chiang. MultiMediaMiner: A System Prototype for Multimedia Data Mining. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 581–583, Seattle, Washington, USA, June 1998. ACM Press.
- [ZKY02] M. Zhang, B. Kao, and C.-L. Yip. A Comparison Study on Algorithms for Incremental Update of Frequent Sequences. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 554–561, Maebashi City, Japan, 2002. IEEE Computer Society.