SEMANTIC-ENHANCED WEB-PAGE RECOMMENDER SYSTEMS

A dissertation submitted for the degree of
Doctor of Philosophy in Computing Sciences

By

Thi Thanh Sang Nguyen

Faculty of Engineering and Information Technology
UNIVERSITY OF TECHNOLOGY, SYDNEY
Australia
December, 2012

CERTIFICATE OF ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Student

Production Note:
Signature removed prior to publication.

ACKNOWLEDGEMENTS

Undertaking and completing a task like this would not have been possible without the encouragement and support of many individuals.

First and foremost, I would like to pay my respects and thanks to my supervisors, Dr. Haiyan (Helen) Lu and Prof. Jie Lu, for their technical ideas and constructive criticism which has significantly contributed to the success of this thesis. Their professional advice has been a great asset to have during my moments of confusion and I truly acknowledge that.

I also acknowledge the great help and cooperation that I received from the staff of UTS library and the School of Software, Faculty of Engineering and Information Technology (FEIT), as well as my colleagues in the Decision Systems and e-Service Intelligence (DeSI) laboratory within the Centre for Quantum Computation and Intelligent Systems (QCIS).

I am grateful to my dear friend, Dr. Tich Phuoc Tran, for his useful advice and friendly support during my PhD candidature.

Last but not least, I would also like to thank my family, who gave me the opportunity for my education and supported me in every direction. Their love and care enabled me to ease my mind and soul so that I could concentrate on this thesis.

Sydney, Australia, December, 2012.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ABSTRACT

This thesis presents a new framework for a semantic-enhanced Web-page recommender (WPR) system, and a suite of enabling techniques which include semantic network models of domain knowledge and Web usage knowledge, querying techniques, and Web-page recommendation strategies. The framework enables the system to automatically discover and construct the domain and Web usage knowledge bases, and to generate effective Web-page recommendations. The main contributions of the framework are fourfold: (1) it effectively changes the fact that knowledge base construction must rely on human experts; (2) it enriches the pool of candidate Web-pages for effective Web-page recommendations by using semantic knowledge of both Web-pages and Web usage; (3) it thoroughly resolves the inconsistency problem facing contemporary WPR systems which heavily employ heterogeneous representations of knowledge bases. Knowledge bases in the system are consistently represented in a formal Web ontology language, namely OWL; and (4) it can generate effective Web-page recommendations based on a set of thoughtfully-designed recommendation strategies. A prototype of the semantic-enhanced WPR system is developed and presented, and the experimental comparisons with existing WPR approaches convincingly prove the significantly improved performance of WPR systems based on the framework and its enabling techniques.

Keywords: Web-page recommender systems, Web usage mining, domain knowledge modelling, knowledge representation, semantic network, semantic reasoning

Chapter 1.

INTRODUCTION

1.1. Background

The explosive growth of information on the World Wide Web with the development of advanced electronic devices has made Web information increasingly important in almost everybody's life. The rapid introduction of current websites has overwhelmed Web users by offering many choices. Consequently, Web users tend to make poor decisions when surfing the Web due to an inability to cope with enormous amounts of information. Recommender systems have proved in recent years to be a valuable means of helping Web users by providing useful and effective recommendations or suggestions. The core techniques in recommender systems are the learning and prediction models which learn users' behaviour and evaluate what users would like to view in the future. In particular, a recommender system can suggest interesting items from a large set of items based on the knowledge gained about an active user (Konstan & Riedl 2012; Ricci, Rokach & Shapira 2011). "Item" in these studies is the general term used to denote what the system recommends to users, which might be a CD, a book, a piece of news, or a Web-page. Today, recommender systems are familiar to everyone who surfs websites on the Internet on a regular basis. There are many examples of recommender systems for commercial websites, such as Amazon and CDNow; movie recommendation sites, such as Moviefinder and Movielens; and even news sites, such as NASA and MSN.

Web-page recommender systems are one kind of recommender systems, which can automatically recommend Web-pages that are most interesting to a particular user based on the user's current Web navigation behaviour. Since a website is usually designed to show the index pages on the home page, the index pages take the role of guiding users to the content pages on the website through Web-page links (Gündüz-Ögüdücü 2010), whereas with the index pages, a user usually has to navigate a number of Web-pages to reach the

content page they are interested in. If the index pages of a website are not well designed, which is often the case, Web users will struggle to find useful pages and are very likely to leave the site. For a commercial website, this means losing potential customers. For an e-government website, this will mean that the citizen's needs are not satisfied. Therefore, Web-page recommender systems have become increasingly valuable for helping Web users to find the most interesting and useful Web-pages on specific websites. Good Web-page recommendations can improve website usage and Web user satisfaction.

How to make effective Web-page recommendations to Web users without excessive input from those users is a hot research topic. Significant effort has been devoted to developing effective Web-page recommender systems; however, a number of challenges or problems, as listed below, have been encountered in the development of contemporary Web-page recommender systems.

"*New page" (cold-start) problem*

The popular approach to Web-page recommendation is based on recommendation rules which are built upon sequential Web access patterns that consist of frequently visited Web-pages. These Web access patterns are usually discovered by applying sequence mining techniques to the Web usage data, which is often obtained from Web server logs.

Web access sequence mining falls into the Web usage mining (WUM) stream, which is a Web mining technique stream (Markov & Larose 2007b) and plays an important role in discovering Web-page navigation patterns from a large collection of Web usage data. The existing tree-based sequence mining algorithms, such as PLWAP-Mine (Ezeife & Lu 2005), and probability models, such as Markov model (Borges & Levene 2005), are commonly used as Web access sequence mining techniques.

With this approach, Web-page recommendations are limited by the fact that the recommended Web-pages can only be the pages that are part of the frequent Web access patterns (Dai & Mobasher 2005). If a user is visiting a Web-page that has not been accessed before, e.g. a newly-added Web-page, the user cannot obtain a recommendation. This phenomenon is often referred to as the "new page" problem. The reasons why such a

phenomenon occurs are threefold: (i) the recommendations are generated based on the recommendation rules obtained from the frequent Web access patterns discovered from the Web usage dataset; (ii) the new page is not included in the Web usage dataset so it cannot appear in any discovered patterns; and (iii) the systems do not have a recommendation rule corresponding to the new page.

Challenges of manual ontology construction

The use of semantic knowledge in recommender systems is blooming because it can offer the opportunity to semantically enrich knowledge bases about Web-pages, and it has therefore become a promising solution for offering effective Web-page recommendations (Dai & Mobasher 2005). The backbone technology for semantic knowledge representation is ontologies. Ontological representation of the semantic knowledge can be machine-understandable and can assist in interpreting, analysing, and reasoning about the Web access patterns discovered in the mining phase (Berendt, Hotho & Stumme 2002; Bose et al. 2007; Khribi, Jemni & Nasraoui 2009; Zhang & Segall 2008). This enables knowledge integration and automated processes by machine (Domingue, Fensel & Hendler 2011; Gündüz-Ögüdücü 2010).

Ontologies in Web-page recommender systems have to date mainly been constructed manually by system developers in consultation with domain experts. Ontology construction is a complex development process which is costly and labour-intensive, and demands a high level of proficiency in the domain (Grimm et al. 2011; Gündüz-Ögüdücü 2010). It is a big challenge to design and construct a perfect ontology for a website because there are usually a huge number of pages on one website and some important concepts in the ontology may be overlooked by developers. Such a challenge may lead to high costs and a long development time, or to incomplete domain knowledge bases.

"Heterogeneous knowledge bases" problem

Several efforts have been devoted to generating semantic knowledge to support Web-page recommendations. However, the knowledge bases used in recommender systems are often implemented in a variety of formats. It is quite possible that some discovered

knowledge is represented by an ontological model and implemented in an ontology language (e.g. OWL [1] Web Ontology Language) in one part of a system while the knowledge bases in other parts are presented in databases and implemented in relational databases, text files or spreadsheets. Such heterogeneous knowledge bases make it almost impossible to share and integrate the knowledge bases to generate more effective Web-page recommendations.

This study proposes a framework for a new semantic-enhanced Web-page recommender system (SWRS) and a suite of techniques to resolve or alleviate the above problems. Based on the framework, an effective Web-page recommender system can be developed to offer Web users the top-*N* most commonly visited Web-pages from the currently visited Web-page. The knowledge bases used in the system, including the website domain and Web usage knowledge bases, are represented by ontological-style semantic networks which can be implemented consistently in a formal Web Ontology Language (OWL).

In the remainder of this chapter, Section 1.2 presents the research questions and objectives of this study, followed by the research significance and contributions of the study in Section 1.3. Section 1.4 outlines the overall structure of the thesis. The author's publications related to this study are shown in Section 1.5.

1.2. Research Questions and Objectives

With the aim of addressing the problems identified in Sub-section 1.1, this study answers the following questions:

- How can we *understand* a user's Web access sequence to effectively recommend the next page request to the user?
- How can we *automatically* discover and represent useful knowledge for Web-page recommendation given the Web usage data?
- How can we *consistently* represent knowledge bases for Web-page recommendation in recommender systems?

[1] http://www.w3.org/2007/OWL/wiki/OWL_Working_Group; Accessed September, 2012

- How can we *incorporate* the discovered knowledge into a Web-page recommender system?
- How can we *enrich* the Web-page candidate pool to include more relevant Web-pages with the Web-page navigation patterns, and
- How can we generate *effective* Web-page recommendations?

In answering these questions, this study has achieved the following five research objectives. The primary objective is stated first, and is followed by four other specific objectives.

Objective 1: Propose a new framework for semantic-enhanced Web-page recommender systems to improve Web-page recommendations

A new framework for semantic-enhanced Web-page recommender systems (SWRS) has been developed**.** This framework enables automated Web usage knowledge discovery and representation and the automated integration of domain and Web usage knowledge in a uniform way to support semantic-enhanced Web-page recommendations. With this framework, we have: (1) discovered Web usage knowledge from Web server log files; (2) applied domain knowledge to interpret the semantics of Web-pages in the Web usage knowledge; (3) applied new techniques to automate the integration of domain and Web usage knowledge using ontology technology; and (4) applied new recommendation strategies to generate recommendation rules to support more effective Web-page recommendation.

Objective 2: Develop two domain knowledge representation models for Web-page recommendation using ontology language

In achieving this objective, this study has developed (i) a domain knowledge discovery process, and (ii) two domain knowledge representation models to support Web-page recommendation. In the domain knowledge discovery process, Web-pages are first discovered from a given website, and domain terms are extracted from the metadata of the discovered Web-pages, specifically, the titles of Web-pages. The two domain knowledge representation models are used to express the semantics of Web-pages, so they include the

extracted domain terms, the relationships between them, the discovered Web-pages, and the associations between the domain terms and the Web-pages. One of the domain knowledge representation models is a domain ontology model which is used to construct a domain knowledge base of Web-pages with the help of expert human knowledge in the domain. Another is a semantic network model which is used to automatically construct a domain knowledge base of Web-pages based on an automatic knowledge discovery process. Both models are developed and implemented in ontology language.

Objective 3: Develop two Web usage knowledge representation models using ontology language

In achieving this objective, this study has developed (i) a Web usage knowledge discovery process, (ii) a first new Web usage knowledge representation model, called a Web-page navigation model, and (iii) a second new Web usage knowledge representation model, called a domain term navigation model. In the Web usage knowledge discovery process, a set of frequent Web access patterns (FWAP) are discovered using an advanced WUM technique. With aid the of domain knowledge, represented by the domain ontology or the semantic network, the domain terms of Web-pages in FWAP are identified so that the set of FWAP are converted into a set of frequently viewed domain term patterns (FVTP). The first Web usage knowledge representation model, i.e. the Web-page navigation model, is proposed to represent a weighted network of FWAP, while the second model, i.e. the domain term navigation model, is proposed to represent a weighted network of FVTP. By using ontology language, a schema is designed for each of the two models to automatically generate a weighted network of Web-page navigation or domain term navigation, respectively.

Objective 4: Develop Web-page recommendation strategies based on the proposed knowledge representation models

In achieving this objective, this study has developed a set of novel recommendation strategies to generate Web-page recommendations given the last one or two Web-pages accessed by a user. The recommendation strategies offered fall into three types: (1) without semantic enhancement, (2) with semantic enhancement based on the domain ontology, and

(3) with semantic enhancement based on the semantic network. Specifically, the first type of strategy uses the Web-page navigation model to predict the next Web-pages to be accessed. The second type of strategy coordinates the Web-page navigation model and the domain ontology-based domain term navigation model to predict the next Web-pages to be accessed. The third type of strategy coordinates the Web-page navigation model and the semantic network-based domain term navigation model to predict the next accessed Web-pages.

Objective 5: Develop a prototype of the semantic-enhanced Web-page recommender system

In achieving this objective, this study has developed a prototype of the SWRS which can be used for a real world website. The domain and Web usage knowledge representation models are involved in carrying out the Web-page recommendation strategies in the system.

1.3. Research Significance and Contributions

By tackling the five stated objectives, this study contributes to the area of Web-page recommender systems in the following aspects:

(1) It develops a new framework of the SWRS consisting of five process stages: Pre-processing, Web usage mining, Domain knowledge construction, Prediction model, and Web-page recommendation generation, which *significantly improves* the performance of Web-page recommendation. The system can alleviate the "*new page*" problem by integrating semantic knowledge with Web usage mining. For example, when a user accesses a new page which is not in the discovered Web usage patterns, but whose domain terms were learned in the domain term navigation network which is generated based on the proposed knowledge representation models, the system can recommend Web-pages through the domain terms of Web-pages.

(2) It proposes two domain knowledge representation models for semantic enhancement. The domain knowledge can *enrich the semantics* of Web-pages by the use of

human knowledge in the domain ontology model, or the domain terms associated with the Web-pages in the semantic network model.

As stated in Objective 2, domain terms in the domain ontology model are generalized to domain concepts to conduct an ontology schema, based on which a domain ontology can be edited with term instances. A mapping algorithm is built to automatically map Web-pages to corresponding term instances in the domain ontology, which will reduce the human labour cost, particularly since the number of Web-pages is very large. This model is applied to a specific website, i.e., the website www.microsoft.com (MS), using Microsoft anonymous Web data, which is available to the public on the Web and is useful for testing Web-page recommendation experiments. Furthermore, the domain ontology model is designed in a *changeable way* which enables the future update of domain ontology with new domain terms and Web-pages.

The semantic network model, on the other hand, is proposed on the basis of the fact that there are collocations of domain terms in Web-page titles. A term extraction algorithm is built to extract sequences of domain terms from Web-page titles to support the construction of the semantic network. A schema of the semantic network model is designed to *automatically* generate a semantic network of Web-pages given the extracted term sequences. This semantic network model is more efficient at solving the high cost of human labour, and can be applied to any website.

Generally speaking, both of these domain knowledge representation models can *efficiently reason* about domain terms, Web-pages, and the relationships among them to semantically enhance Web-page recommendation.

(3) It proposes two Web usage knowledge representation models for Web-page or domain term-based prediction.

As stated in Objective 3, this study first discovers the most efficient tree-based sequential pattern discovery technique, and then applies it to mine Web usage data and to generate a complete set of FWAP. The rationale for using this approach is that the *most popular Web-pages* can be recommended. In the absence of more precise information about

a user's preferences, a frequently visited page, i.e., something that is liked by many users, will probably also be liked by a generic user. Hence, the utility of these frequently visited pages is predicted to be reasonably high for the generic user. Furthermore, FVTP, which is the result of the integration of the domain knowledge and FWAP, is useful for predicting the users' next requests *through domain terms*.

These sets of frequent patterns, i.e., FWAP and FVTP, are then transformed into types of *weighted networks* which enable them to predict the next event given the occurrence of a few events with estimated probabilities. These weighted networks of FWAP and FVTP are *automatically* generated by using the proposed Web-page model and domain term navigation model, respectively, and are represented in the ontology language. Thus, the two models can co-operate with each other through the domain ontology or semantic network model and can efficiently make Web-page recommendations.

(4) It proposes a set of novel Web-page recommendation strategies that coordinate the knowledge representation models. The proposed models have been developed *in agreement with each other* and *ontologically* described using OWL, so the built knowledge is *machine-understandable* and *machine-processable*, so they can be *easily integrated* with each other to facilitate the generation of Web-page recommendations.

(5) It validates the knowledge representation models, and examines the Web-page recommendation strategies in the SWRS by conducting a set of experimental cases on public and real world datasets. The public data is the Microsoft anonymous Web data from the UCI machine learning repository, and the real world dataset is a Web log of the handbook website of the University of Technology, Sydney (UTS). Another experimental case using the best sequence mining technique for Web-page recommendation is conducted as the base case for comparison with the experimental cases, using the proposed strategies, in terms of Web-page prediction accuracy.

Extensive empirical analyses conducted on the data used suggest that the strategies using the proposed models are *effective* solutions for Web-page recommender systems since the proposed strategies *significantly outperform* the base recommendation strategy. The

semantic network-based recommendation strategies, especially, are able to achieve the highest performance. This also validates the SWRS, using the proposed models.

(6) It develops a prototype for the SWRS which includes back-end components with user interfaces for controlling the system, and a front-end component in the form of a Web browser with which a user can access Web-pages through the system. The prototype is a component-based system which is flexible enough to be extended in the future; that is, we can improve a single component by using a new method for the process in the component as long as the compatibility of the input and output data throughout the process is ensured. For example, we can use a different WUM technique, e.g. association rules, or a tree-based pattern discovery method, in the WUM process, or we can improve domain knowledge construction in some way.

1.4. Organization of the Thesis

This thesis is organized into nine chapters which include an extensive review of relevant Web mining and ontology techniques in the context of recommender systems, followed by the proposed novel knowledge discovery and representation methods, their applications to Web-page recommender systems, and the implementation of the SWRS. In particular, the first two chapters provide the background to Web-page recommendation problems, the proposal, and related works. Chapter 3 investigates sequence mining techniques for the purpose of selecting the best algorithm for Web usage mining in the SWRS. The following five chapters cover research methodologies and the use and implementation of the proposed models in the SWRS. Chapter 9 summarizes key findings and future research directions. Figure 1-1 serves as a map of this thesis and shows the logical relationship of all the chapters in the thesis. A descriptive list of chapters follows.

Chapter 1 introduces the topic of this research, that is, the semantic-enhanced Web-page recommender systems using novel knowledge representation models. This chapter opens with a description of the background to the topic and then states the research objectives, and the main contributions of the thesis.

Chapter 2 presents a literature review of related areas, including Web mining techniques, ontology technology and recommender systems. Firstly, a brief summary of the background of Web mining techniques, such as Web content mining, Web structure mining and Web usage mining, is provided. Secondly, background knowledge of ontology, such as definitions, roles of ontology, ontology models, ontology languages, variants of ontologies, ontology construction and other related issues, is discussed. Finally, this chapter reviews recommender systems in general and Web-page recommender systems in particular.

Chapter 3 investigates and compares sequential pattern mining techniques for Web usage to establish the best Web usage mining algorithm for pure usage-based Web-page recommender systems.

Chapter 4 presents the domain ontology modelling of a website for the SWRS. Firstly, the domain ontology model of a website is presented. Secondly, a new method for domain ontology modelling is proposed, and reasoning algorithms for the domain ontology model are described. Thirdly, a case study of the development of a domain ontology of the MS website is shown. Finally, this chapter gives an evaluation and discussion of the built domain ontology.

Chapter 5 proposes the model of automatic semantic network construction of a website for the SWRS. Firstly, a new method for the automatic semantic network construction of Web-pages is proposed. Secondly, details of the semantic network model, along with reasoning algorithms and an experimental example of the semantic network, are presented. Finally, an experimental evaluation and remarks are detailed.

Chapter 6 proposes the new concept navigation models for prediction. Firstly, the Web-page navigation model is built to automatically generate a Web-page navigation network, and reasoning algorithms for this model are also presented. Secondly, the domain term navigation model is proposed to automatically generate a domain term navigation network, and reasoning algorithms for this model are also presented. Finally, an example of the proposed navigation models is illustrated.

Chapter 7 proposes a new framework for the SWRS, coordinating the proposed knowledge representation models to make Web-page recommendations. Firstly, the new framework of the SWRS is presented. Secondly, the three types of novel Web-page recommendation strategies are proposed. Thirdly, the experimental evaluation of the SWRS using the Web-page recommendation strategies on the public and real world datasets is shown. Finally, some discussions are presented.

Chapter 8 presents a prototype of the SWRS based on the proposed framework of the SWRS and enabling techniques. The system architecture comprising the back-end processing components and the front-end component is demonstrated and the back-end processing components are developed as sub-systems in the system. The front-end component is a Web user interface displaying Web-page content and Web-page recommendations. The data structures of the back-end sub-systems are then explained. Lastly, the operation and interfaces of the SWRS are described as practical guidelines.

Chapter 9 summarizes the thesis and draws conclusions. Possible improvements and several future research directions are then discussed.

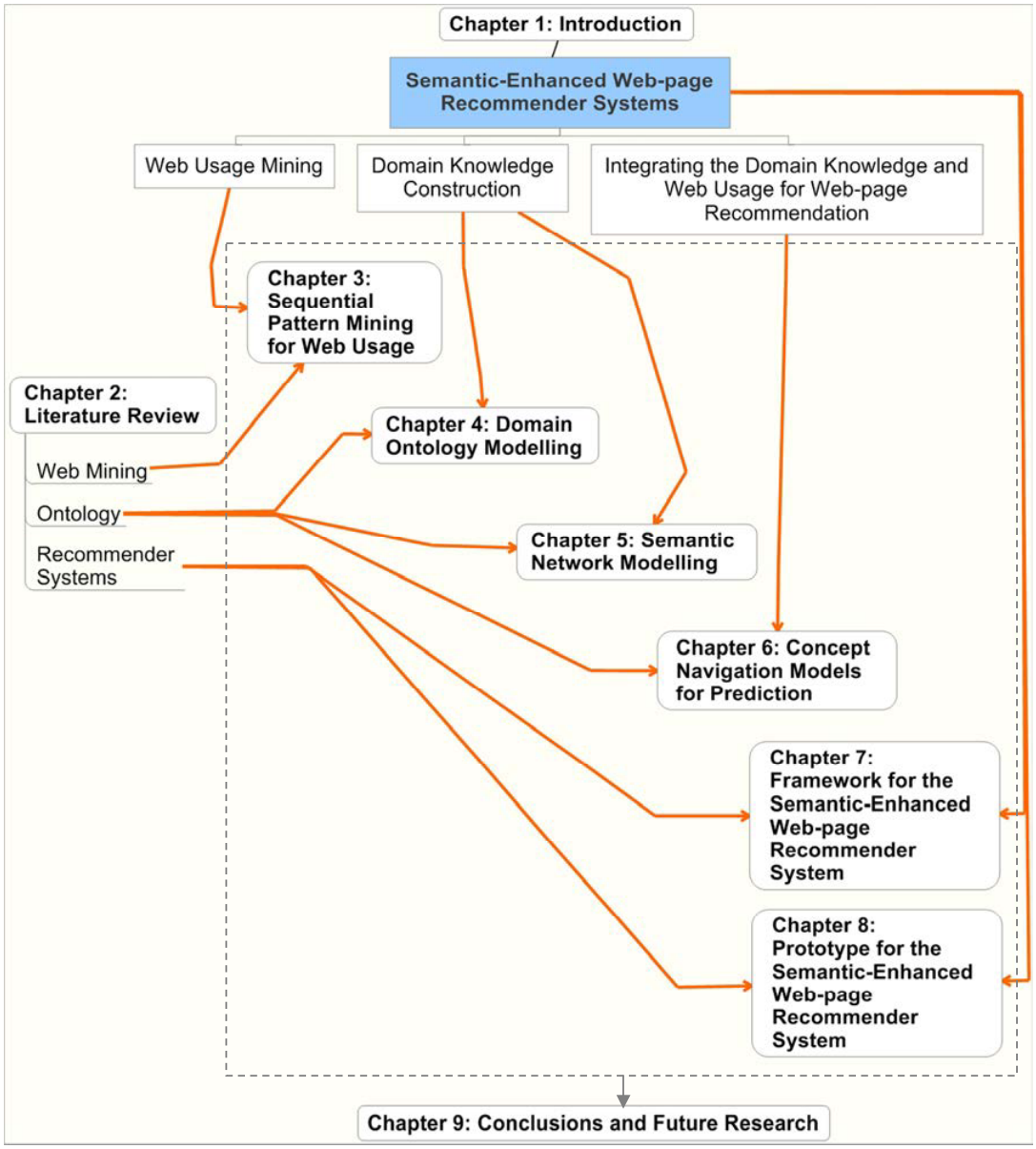


Figure 1-1: The overall structure of the thesis

1.5. Publications Related to the Thesis

Journal Papers:

1. **Nguyen**, T.T.S., Lu, H., Tran, T.P. & Lu, J. 2012, 'Investigation of Sequential Pattern Mining Techniques for Web Recommendation', paper accepted for publication in *International Journal of Information and Decision Sciences (IJIDS).*

2. **Nguyen**, T.T.S., Lu, H. & Lu, J. 'Semantic-Enhanced Web-page Recommendation', paper submitted to *the IEEE Transactions on Knowledge and Data Engineering* (ERA tier-A journal), and under major revision.

3. **Nguyen**, T.T.S., Lu, H. & Lu, J. 'Automating Construction of an Ontology-based Semantic Network of Pages given Web Usage Data', paper submitted to *Data Mining and Knowledge Discovery* (ERA tier-A journal).

4. **Nguyen**, T.T.S., Lu, H. & Lu, J. 'A Semantic-Enhanced Web-page Recommender System based on Concept Navigation Models, paper submitted to *IEEE Intelligent Systems* (ERA tier-A journal).

Conference Papers:

5. **Nguyen**, T.T.S. 2009, 'Efficient Web Usage Mining Process for Sequential Patterns', paper presented to the *11th International Conference on Information Integration and Web-based Applications & Services (iiWAS)*, 14-16 December, Kuala Lumpur, Malaysia, pp. 465-469.

6. Lu, H., **Nguyen**. T.T.S. 2009, 'Experimental Investigation of PSO Based Web User Session Clustering', paper presented to the *1st International Conference on Soft Computing and Pattern Recognition*, 4-7 December, Malacca, Malaysia, pp. 647-652.

7. **Nguyen**, T.T.S., Lu, H. & Lu, J. 2010, 'Ontology-Style Web Usage Model for Semantic Web Applications', paper presented to the *10th International Conference on Intelligent Systems Design and Applications (ISDA)*, 29 November-1 December, Cairo, Egypt, pp. 784-789.

Chapter 2.

LITERATURE REVIEW

2.1. Introduction

The theme of this literature review is semantic-enhanced recommender systems. The literature review covers the background, latest development of and related techniques for semantic-enhanced recommender systems. Figure 2-1 shows the topic map in this review, where Web mining and ontology are the two pillar techniques to semantic-enhanced recommender systems, and evaluation metrics are the core part to make sure that the techniques are useful for semantic recommender systems. Web mining techniques are used to discover useful patterns from Web data. They can be classified into three streams as Web content mining, Web structure mining and Web usage mining. Web usage mining stream is the focus of Web mining in this study. Ontology is an advanced knowledge organization technique as the backbone of the Semantic Web technology. The definitions, representation languages, issues about ontology construction, learning and reasoning are highlighted in the context of Web-page recommender systems. The standard evaluation metrics are also summarized.

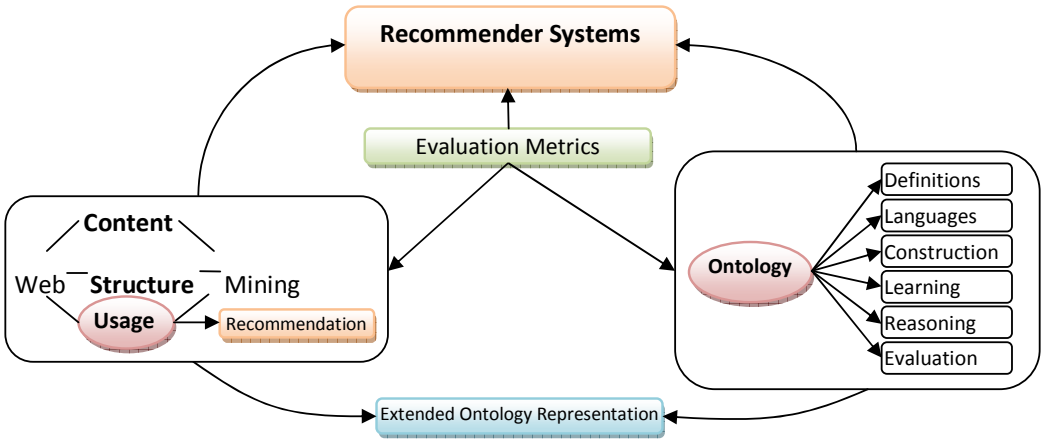


Figure 2-1: Topic map

2.2. Web Mining

Web mining (WM) is the process of discovering useful knowledge from Web data. Depending on different types of Web data, appropriate mining techniques are selected. There are three main broad categories of Web mining (Kolari & Joshi 2004).

- Web content mining (WCM) is used to mine Web content, such as HTML or XML documents.
- Web structure mining (WSM) focuses on Web structure, such as hyperlinks on Web-pages.
- Web usage mining (WUM) is applied to Web usage data, such as Web logs or clickstreams, from a website. This category will be presented more deeply in this section.

2.2.1. Web Content Mining

In Web content mining, clustering and classification are main mining techniques applied to Web documents (Markov & Larose 2007b). Clustering is a kind of *unsupervised learning*, used to group Web documents or to organize them in the hierarchies of their topics. Clustering can be applied to a set of objects which are presented by attributes and values, called the *value* (or *feature-value*) *representation*. On the other hand, the general framework for classification is usually called *supervised learning*. Classification aims to map a set of documents to a set of class labels. In both cases, the common type of document representation is the *vector space model* where the features are the terms of the document.

2.2.2. Web Structure Mining

Web structure mining discovers useful knowledge from the hyperlink structure of the Web. It is often combined with Web content mining to retrieve a set of Web-pages based on the links. It is usually used for PageRank, Web search, or Web crawling (Liu 2011b; Markov & Larose 2007b; Stumme, Hotho & Berendt 2006). From a link of a website, for example, a Web crawler can follow hyperlinks to collect linked pages at different levels. Web crawlers

(Menczer 2011) can be applied to business intelligence, whereby organization want to collect their competitors' information; or monitoring websites, and pages of interest in order to notify new information appearing in certain places to Web users.

A basic crawler algorithm (Menczer 2011) initializes with a set of seed pages, then uses the links within them to fetch other pages. The links in these pages are, in turn, extracted to visit the corresponding pages. The process repeats until a sufficient number of Web-pages are achieved. Another Web crawling method for Web forum, stated in (Yan et al. 2006), extracts board page seeds from a homepage, then get a link queue including all subsequent board pages in the same board with each board page seed. Pages in each link queue are downloaded, and links of post pages are extracted from each page which is a board page. Finally, all post pages in all board pages are downloaded. Rather than crawling pages from the entire website, we may want to crawl Web-pages in certain categories or topics of interest, which is a focused crawler. In this case, similarity measures, such as cosine similarity based on Term Frequency (TF) or Term Frequency–Inverse Document Frequency (TF-IDF) (Liu 2011a), might be used to estimate content similarity.

2.2.3. Web Usage Mining

Web usage mining aims to discover some useful patterns from the Web usage data, such as, clickstreams, user transactions and users' Web access activities, which are often stored in Web server logs (Liu, Mobasher & Nasraoui 2011). A Web server log records user sessions of visiting Web-pages of a website day by day. It can be used to discover potentially useful Web usage knowledge, e.g. the navigational behaviour of Web users, (Mobasher 2007a). Generally speaking, a Web usage mining process includes three phases: pre-processing, mining, and applying mining results (Woon, Ng & Lim 2005). After pre-processing Web log files, Web access sequences (WAS), for example, are generated and filed in a dataset (Ezeife & Liu 2009). An element of this dataset is a sequence of representing a user browsing session. In the mining phase, some sequential pattern mining techniques, such as clustering, classification, association rules, and sequential pattern discovery (Pierrakakos et al. 2003), can be applied to the WAS to extract the frequent Web access patterns (FWAP), which is useful Web usage knowledge. In the third phase, the discovered knowledge will be

used in a specific application, e.g., a Web-page recommender system, in which FWAP are used for generating the recommendation rules to support on-line Web-page recommendation. The mining phase using sequential pattern mining techniques is the core phase in a WUM process and plays a crucial role in a Web-page recommender system to support users to make better decision based on their current Web navigation history.

Some applications of WUM include (i) to cluster Web users based on interesting patterns (Chen, Bhowmick & Li 2006), to mine conceptual link hierarchies from Web log files for adaptive website navigation (Zhu, Hong & Hughes 2004), (ii) to build the FWAP using a tree algorithm (Ezeife & Lu 2005), and (iii) to predict Web navigations using the Markov model or association rules (Khalil 2008; Liu, Huang & An 2007). Predicting Web access patterns (WAP) using the Markov model is very interesting in Web personalization because of its special features, such as, modelling a collection of navigation records, modelling user Web navigation behaviour, or classifying browsing sessions into different categories.

According to J. Borges and M. Levene (2004), the Markov model is a powerful and probabilistic model to estimate the probability of visiting Web-pages. Each Web-page is referred to as a state in the Markov model. In particular, the N-order Markov model can predict the next visited page based on the previous N-1 visited pages. The predictive probability of the N-order Markov model is generally higher than the lower-order model, however, the number of states used in a higher-order Markov model will increase exponentially. Because the model complexity is measured by the number of states, the complexity of a higher-order Markov model excessively increases when using it to model a huge number of Web-pages. Hybrid probabilistic predictive models based on the Markov model, such as the dynamic clustering-based Markov model of J. Borges and M. Levene (2004), have shown improved prediction accuracy over the traditional Markov model. However, the complexity of the Markov-based models has caused concerns when they are used in Web-page recommender systems because there are usually a great number of pages in a website. One effective way to reduce the complexity of a Markov-based model is to filter out the insignificant Web-pages in the Web usage data. Some clustering methods have been applied to filter Web-pages, such as Expectation–Maximization (EM) and k-means

algorithms in (Khalil 2008). Moreover, by combing with link analysis, Eirinaki and Vazirgiannis (2007) assign probabilities to the accessed Web-pages based on their importance in the website's navigational graph and build a hybrid probabilistic predictive model based on Markov models for Web-page ranking and recommendation. This approach achieves more objective and representative predictions, and provides ranked recommendations to Web users.

On the other hand, the tree-based algorithms are also contributed significantly to modelling the navigational activities in websites (Liu, Mobasher & Nasraoui 2011; Zhou 2004). With this approach, an aggregate tree structure is built to model the navigational trails in session data. Each node in the tree represents a navigational subsequence from the root (an empty node) to a page, and labelled by the page and the frequency of occurrences of that subsequence in the session data. The advantage of this approach is that it is very efficient to search navigational patterns, and easy to obtain the support and confidence of navigational patterns. The support and confidence of navigational pattern is computed based on its occurrence frequency in the navigational sequences (Liu, Mobasher & Nasraoui 2011). However, the disadvantage of this approach is the high level of space complexity, especially for the websites that have a great number of Web-pages. Recent literature has shown that the WAP-tree based algorithms outperforms the other sequential pattern mining algorithms, e.g. Apriori-based algorithms, and pattern-growth based algorithms, in terms of speed and memory (Mabroukeh & Ezeife 2010). This explains the high popularity of the WAP-tree based approach, such as pre-order linked WAP-tree mining (PLWAP-Mine) and conditional sequence mining (CS-Mine), in Web-page recommender systems (Ezeife & Liu 2009; Wang, Bai & Li 2010; Zhou, Hui & Chang 2004).

In summary, making recommendations is the main application of WUM in recommender systems as WUM can obtain the actual user behaviour rather than the behaviour expected from the Web design. And pattern discovery methods play an important role in mining user behaviour sequences.

2.3. Ontology

The Internet has become very popular nowadays. Millions of people access the Web to search information, do online shopping, get entertained or just learn. From its early stages, the Web has provided a magnificent opportunity for anyone: persons, businesses or communities that want worldwide exposure. However, communication between machines has not been developed deeply enough. The Internet before 1980s did not allow for fluent communication between machines to do anything more but searching for words, whereas they should be exchanging information about viewed Web-pages. Overcoming, or at least lowering, existing barriers to more efficient and automatic human-machine communication is at the forefront of research and development efforts.

The Semantic Web has been proposed by Tim Berners-Lee in 2000s, as an extension of the current Web, namely a machine-readable Web which facilitates human-computer cooperation (Domingue, Fensel & Hendler 2011). The vision of the Semantic Web is to enable machines to interpret, understand, and process information in the World Wide Web in order to respond users' requests (Henze, Dolog & Nejdl 2004). Ontology has been considered as the backbone of the Semantic Web technology for representing and sharing knowledge between Web applications since 1980s. This study develops knowledge representation models for domain and Web usage knowledge using ontology technology. This section will first list various definitions of an ontology, briefly state the roles of ontology in applications, and present ontology languages, variants of ontologies, and examples of ontologies. It then describes the issues of ontology construction, ontology learning, and ontology reasoning. Finally, it lists some methods for ontology evaluation.

2.3.1. Definitions of an Ontology

An ontology may include individuals (instances), classes, attributes, relations, restrictions, rules, and axioms. Based on these components, we can build an object-oriented model for an application domain and use this model for sharing and reusing domain information on the Web. Such an ontology model allows human- and machine-understandable content and human-machine interaction. The following presents definitions of an ontology.

According to Gruber (Gruber 1995), ontology is a formal representation of the world. It defines a set of representational primitives that are relevant for modelling a domain of knowledge or discourse. The representational primitives typically consist of a set of concepts (or entities within a domain), relationships (that may exist among these concepts), and properties (or attributes that distinguish each concept).

Definition 2.1 *(Ontology)* (Antoniou & Harmelen 2008) According to T.R. Gruber's definition, later refined by R. Studer, "*An ontology is an explicit and formal specification of a conceptualization*".

Definition 2.2 *(Ontology)* (Guarino, Oberle & Staab 2009) An ontology is defined as follows:

"Let **C** *be a conceptualization, and* **L** *is a logical language with vocabulary* **V** *and ontological commitment* **K**. *An ontology* $\mathbf{O_k}$ *for* **C** *with vocabulary* **V** *and ontological commitment* **K** *is a logical theory consisting of a set of formulas of* **L***, designed so that the set of its models approximates as well as possible the set of intended models of* **L** *according to* **K**.*"*

In which, "A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose."

Definition 2.3 *(Ontology)* (Grimm et al. 2011) An ontology is a tuple $\boldsymbol{O}$ = (**S**, **A**), where **S** is a signature, and **A** is a set of axioms. The signature comprises several sets $\mathbf{S} = \mathrm{C} \cup \mathrm{I} \cup \mathrm{P}$ of classes C, instances I and properties P. The signature entities are further divided into sets $\mathrm{C} = \boldsymbol{\mathcal{C}} \cup \boldsymbol{\mathcal{D}}$, $\mathrm{I} = \boldsymbol{\mathcal{I}} \cup \boldsymbol{\mathcal{V}}$, $\mathrm{P} = \boldsymbol{\mathcal{R}} \cup \boldsymbol{\mathcal{T}}$ of concepts $\boldsymbol{\mathcal{C}}$ and datatypes $\boldsymbol{\mathcal{D}}$, individuals $\boldsymbol{\mathcal{I}}$ and data values $\boldsymbol{\mathcal{V}}$, and relations $\boldsymbol{\mathcal{R}}$ and attributes $\boldsymbol{\mathcal{T}}$. Axioms can fall into some types: Instantiation, Assertion, Subsumption, Domain, Range, and Disjoitness.

Besides, Wang, Liu & Yu (2007) define an ontology as a 4-tuples including a Term Set (*T*), an Individual Set (*X*), a Term Definition Set (*TD*), and an Instantiation Assertion Set (*XD*) for information retrieval applications.

Lu et al. (2009) describe an ontology model including *C* (an instance set of Concepts), *O* (Operation set), *P* (Participants), *R* (Rules): *P* x *C* → *O*.

Informally speaking, an ontology is a conceptual model that specifies the terms and relationships between them explicitly and formally, which in turn represent the knowledge for a specific domain (Antoniou & Harmelen 2008). There should be three main components in an ontology, as follows (Boyce & Pahl 2007):

- Domain terms (concepts),
- Relationships between the terms (concepts), and
- Features of the terms and relationships.

The terms refer concepts (classes of objects) in the domain. The relationships include hierarchies of concepts (classes). For example, a domain ontology for personalized e-learning in educational systems (Boyce & Pahl 2007) can be illustrated as Figure 2-2, which includes classes, data type properties and relationships.

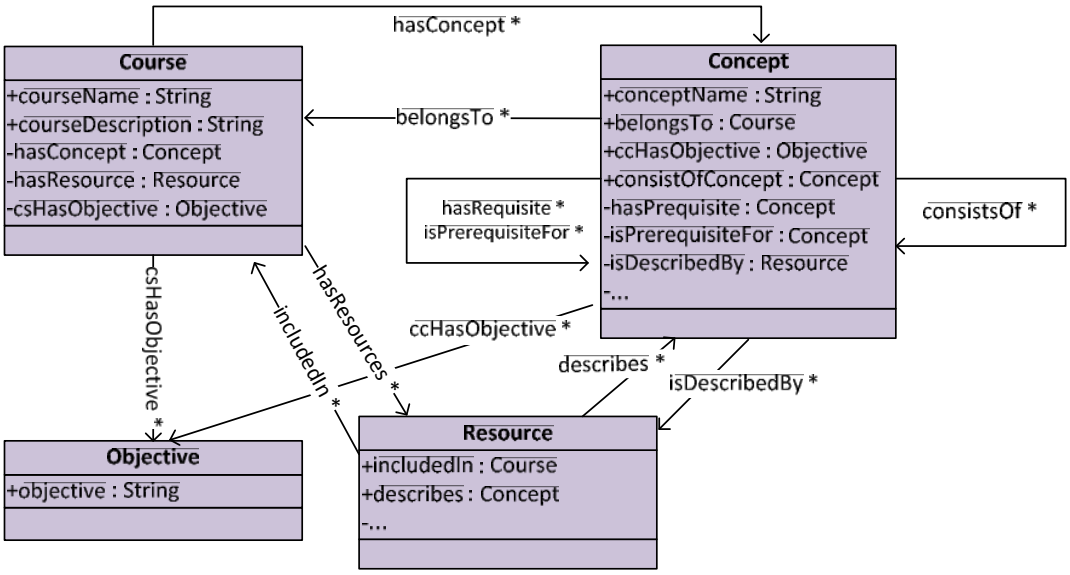


Figure 2-2: Excerpt of a domain ontology for personalized e-learning in educational systems (Boyce & Pahl 2007)

In this example, class Course represents the subjects being taught in an educational application. The class contains properties representing the features of a course, i.e., *courseName* and *courseDescription* are the name and a brief description of the course. The relationships between a course and other instances: *csHasObjective* points to the objectives which are defined by class Objective, whereas *hasConcept* and *hasResource* point to a set of concepts and resources, respectively, that compose a course. Class Concept contains a data type property *conceptName*, to identify the concept, and other object properties that allow to establish different relations among domain concepts: (1) *consistsOf* serves to define a concept hierarchy, and therefore, to establish a relations among a concept and its sub-concepts, (2) *hasRequisite* and *isPrerequisiteFor* (its inverse) point to concepts that must be learned before starting to study a concept, and the concepts for which it is a prerequisite, respectively. On the other hand, a concept points to a collection of objectives via the object property *ccHasObjective*. In addition, the object property *isDescribedBy* (in class Concept) points to digital resources which can be included in several courses (object property *includedIn – hasResource* is its inverse).

In overall, ontology serves as a useful mean to facilitate representing concepts and relations between them in different domains. Ontology has been popular in knowledge representation areas because of the following main roles.

2.3.2. Roles of Ontology

The main role of ontology is to represent knowledge in a format which is machine understandable and to support knowledge sharing. The actual contribution of ontology to access and use of Web resources is summarized by Gruninger and Lee as follows (Bürger & Simperl 2010; Hepp 2008). Ontologies can be used:

- "For communication
  - Between implemented computational systems
  - Between humans
  - Between humans and implemented computational systems
- For computation inference
  - For internally representing plans and manipulating plans and planning information

  - For analysing the internal structures, algorithms, inputs and outputs of implemented systems in theoretical and conceptual terms.
- For reuse (and organization) of knowledge
  - For structuring or organizing libraries or repositories of plans and planning domain information".

In the context of Web-page recommender systems (Gündüz-Ögüdücü 2010; Salin & Senkul 2009), ontology concepts are used to semantically enhance Web usage data. In such recommender systems, ontology is designed in appropriate ontology models for the knowledge representation of specific domains.

2.3.3. Ontology Languages

Based on XML[2] syntax, ontology languages have been developed for semantic knowledge management. Because XML only serves for exchanging data between parties who have agreed to the definitions beforehand, it is not enough to express the meaning of concepts and to organize the relations between concepts in an application domain. Therefore, it is necessary to use Web ontology languages which can support the richer expressiveness. Generally speaking, Web ontology languages is a collective term for the following languages (Antoniou & Harmelen 2008):

- XML which can be used to provide a surface syntax for structured documents but there is no semantic constraint on the meanings of these documents;
- XML Schema[3] which presents the structure of XML documents;
- RDF[4] which describes a data model of objects (resources) and relations between them, presents a simple semantics for the data model in XML syntax;
- RDFS[5] which is a vocabulary description language for describing properties and classes of RDF resources, and presenting generally semantic hierarchies of such properties and classes; and

[2] http://www.w3.org/XML/, Accessed September, 2012
[3] http://www.w3.org/XML/Schema, Accessed September, 2012
[4] http://www.w3.org/RDF/, Accessed September, 2012
[5] http://www.w3.org/TR/rdf-schema/, Accessed September, 2012

- OWL[6] which is a richer vocabulary description language for describing properties and classes.

OWL is a main Web ontology language which satisfies the requirements of building a domain ontology, including a well-defined syntax, a well-defined semantics, efficient reasoning support, sufficient expressive power, and convenience of expression (Antoniou & Harmelen 2009). Compared with RDF/XML, OWL extends the *trade-off between expressive power* and *efficient reasoning support*. Ontologies are often implemented in a logic-based language, such as OWL/RDF, to become understandable to software agents or software systems. Therefore, ontology-based knowledge representation allows sharing and inter-changing semantic information among Web systems over the Internet. It also enables the reuse of the domain knowledge, and reasoning the semantics of Web-pages from the existing facts (Harth, Janik & Staab 2011) in semantically enhanced recommender systems. More information on OWL languages can be found at http://www.w3.org/TR/owl2-overview/.

In addition, for querying purposes in ontologies, there are two major languages available, i.e. XML query language and SPAROL query language (http://www.w3.org/TR/rdf-spaql-query/), which infers rules based on matching graph patterns to retrieve classes, properties, instances or schema information. The former is complicated for querying RDF documents. The latter is a better candidate for querying RDF as recommended by W3C. Some other query languages are RDQL and SQL.

2.3.4. Variants of Ontologies

According to Grimm et al. (2011), there are many different forms of conceptual models interpreted as ontologies in different dimensions. This sub-section summarises varying forms of ontology appearance, scope, and degree of formality.

Varying form of appearance

Ontologies can appear in some graphical or formal visualization, but are encoded in an ontology language which enables machine-processable. Graphical appearance of an

[6] http://www.w3.org/TR/owl-features/, Accessed September, 2012

ontology might be semantic networks with interlinked conceptual nodes, or the taxonomic hierarchy of domain concepts and the customary relations between them. However, this graphical appearance cannot express complex axioms in ontologies. Ontology languages like OWL can precisely define the meaning of an ontology in terms of logic. Therefore, an ontology often appears as a set of logical formulas representing a set of axioms formalizing the represented knowledge in the ontology. For storage on disk or on the Web, an ontology needs to be expressed in some machine-processable serialization format, such as RDF/XML.

Varying scope

According to the kind of represented knowledge, ontologies may be classified in two types: *upper-level* and *domain* ontologies. Upper-level ontologies, also called top-level ontologies, describe general concepts that can be used across many domains. They are often employed to design the knowledge at the abstract level. Domain ontologies represent specific concepts in the domain of interest.

Varying degree of formality

An ontology's degree of formality concerns about axiomatization in domain knowledge representation. If no or only few axioms constrain the use of the entities in an ontology, it is a *lightweight* ontology. Otherwise, if an ontology includes many axioms constraining the use of nearly all entities and supporting reasoning about them, such an ontology is a type of *heavyweight* ontologies. Heavyweight ontologies are richer in expressiveness, but they are harder to manage. While lightweight ontologies are less restrictive, they are usually wider acceptable, which is very important for knowledge sharing and reuse. There are popular forms of an ontology, such as thesauri, concept schemes, taxonomies, conceptual data models, rule and fact bases, and general logical theories.

Thesauri organize the words used in a certain domain, and restricted to lexical relations between them, such as synonymity or homonymity. The logic-based representation in thesauri is rather low. A well-known example of thesauri is WordNet.

Concept schemes are expressed as informal semantic networks with very limited axiomatization. This ontology form is suitable for collaborative tagging activities in a Web context, such as social networking websites.

Taxonomy is a way of classifying or categorizing a set of vocabulary terms into a hierarchical structure, like a tree, with the root of the tree is the most general category. Each node presents a term and is connected to others via links as parent-child relationships (Boyce & Pahl 2007).

Conceptual data models are often used to express the data structure of a domain in a software system, such as entity relationship diagrams or UML diagrams, by means of classes, attributes of the classes, and association relations between the classes.

Rule and fact bases, e.g. logic programming rule bases, serve as data-intensive knowledge bases tailored toward dealing with some basic reasoning over a large number of individuals. Typically, these ontologies express a rule-based derivation of facts by means of logic programming mechanisms.

General logical theories are the most formal expressivity of an ontology, since the domain of interest is expressed with a rich axiomatization by means of a logic-based knowledge representation formalism, such as first-order predicate logic.

2.3.5. Examples of Ontologies

In recent years, ontologies have attracted widespread attention in constructing domain knowledge of news, software, courses, and e-learning. An ontology is used to store background information in news stories and contributes to the identification of relevant news stories for investigation (Drury & Almeida 2009). An ontology can formalize common concepts in the software engineering realm, such as data, interfaces, classes and methods. This is useful to facilitate some development and management tasks related to software components in application servers and Web services (Oberle, Grimm & Staab 2009). Moreover, several domain ontologies are developed for learning courses in educational systems. For example, domain ontologies of e-learning courses have been implemented to support distance students (Dzemydiene & Tankeleviciene 2008; Gascuena,

Fernandez-Caballero & Gonzalez 2006). While, Dzemydiene and Tankeleviciene (2008) used ontology to present the taxonomical structures of software products under considering different curriculum levels for the distance learning course "E-learning technologies". Gascuena, Fernandez-Caballero and Gonzalez (2006) were interested in the ontological description of learning materials including concepts, resources and software and hardware devices for a study course.

With semantic expressive power, ontology can be used to represent not only domain knowledge, but also other extended knowledge. By discovering the relation between Web access activities, and the temporal and event attributes in Web logs, a Web usage ontology is generated to represent personalized usage knowledge (Zhou, Hui & Fong 2005). On the other hand, available contextual information from the recommended items and the recommendation process in a recommender system are also able to be described by ontology in a proposed semantics-based approach (Loizou & Dasmahapatra 2006). This approach allows the system to make up-to-date recommendation at run-time. In other words, the quality of recommender system is improved considerably thanks to ontology technology.

2.3.6. Ontology Construction

According to (Chen & Chuang 2008), traditional methods of ontology construction can be categorized into four approaches: dictionary-based, text clustering, association rule-based, and knowledge-based. The dictionary-based approach uses a dictionary like WordNet to define concepts and relations for domain ontology construction (Hu et al. 2006). This approach cannot achieve high performance without being combined with another approach and the size of the ontology is restricted to the amount of vocabularies contained in the dictionary. The text clustering approach (Di Martino & Cantiello 2009) is to cluster terms according to their synonyms, in which terms are selected based on their frequency on documents. The resulting cluster model is a hierarchical binary tree for ontology extraction. However, the accuracy of the text clustering is not high because viewpoints on the same word are discrepant from different users and selecting terms is often not easy. On the other hand, the association rule-based approach is to use association rules to build concept

hierarchical trees. The knowledge-based approach (Alani et al. 2003) aims to construct an ontology using the built knowledge bases including basic rules and simple examples in specific domain, but the scope of produced ontologies is restricted by the knowledge bases.

Systematically, Grimm et al. (2011) has stated a generic methodology of ontology engineering comprises three steps.

(1) Requirements analysis: the engineer or domain expert is involved to analyse the requirements of the underlying application scenario, then to describe them by means of an ontology requirement specification document. The description of the requirement should contain information about the scope of the ontology or the level of expressivity.

(2) Conceptualization: the target domain of interest, including the chosen ontological entities and the formulated axioms, is presented in terms of semantic vocabulary and statements. The engineers or domain experts design the basic structure of the ontology for meeting the aforementioned requirements specification. According to Uschold and Gruminger (1996), there are three possible approaches to develop the class hierarchy or the taxonomic relationships as follows:
   - A top-down development process starts from the most general concepts in the domain and then identifies the subsequent specialization of the general concepts.
   - A bottom-up development process starts from the most specific concepts as the leave nodes in the concept hierarchical structure/tree structure, and then groups these most specific concepts into more general concepts.
   - A hybrid development process is the combination of the top-down and bottom-up approaches. The core concepts are first identified in the domain and then generalised and specified appropriately.

(3) Implementation: this specification is explicitly formalized by an appropriate ontology language. In this phase, some automatic approaches to ontology acquisition or reuse might be required.

Although the above approaches are useful to build an ontology, they might have to construct the ontology manually or semi-automatically. The quality of ontologies relies heavily on the ontology developers' knowledge. Therefore, ontology construction from these approaches is a challenging and time-consuming endeavour.

Recently, some studies have proposed approaches to automatic domain ontology construction using the Bayesian network (Chen & Chuang 2008; Park, Han & Choi 1995). By using the Bayesian network, terms are connected each other with weights calculated based on their frequencies in Web documents, and then an ontology can be built given the terms and relations represented in the network. The ontology development has been improved in terms of reducing the reliance on human experts and development effort. Drury and Almeida (2009) have proposed a method of producing an ontology from news stories by using a recursive algorithm. The recursive algorithm is responsible for identifying relevant news stories from a corpus. In order to refine the built ontology, this method also describes a pruning procedure of removing extraneous information from the ontology. Although this method was not perfect to remove all erroneous information, it can quickly generate a rich and detailed domain ontology of news.

The next sub-section will discuss about ontology learning which refers to the automatic or semi-automatic generation of ontologies from various kinds of data sources.

2.3.7. Ontology Learning

Ontology learning is a process of learning of new ontologies from some available resources, as well as extending the structures of existing ontologies to enrich these ontologies (Sosnovsky 2008). A general ontology learning architecture for the Semantic Web was proposed in (Maedche 2002), as shown in Figure 2-3. It is a process of ontology import, extraction, pruning, refinement, and evaluation.

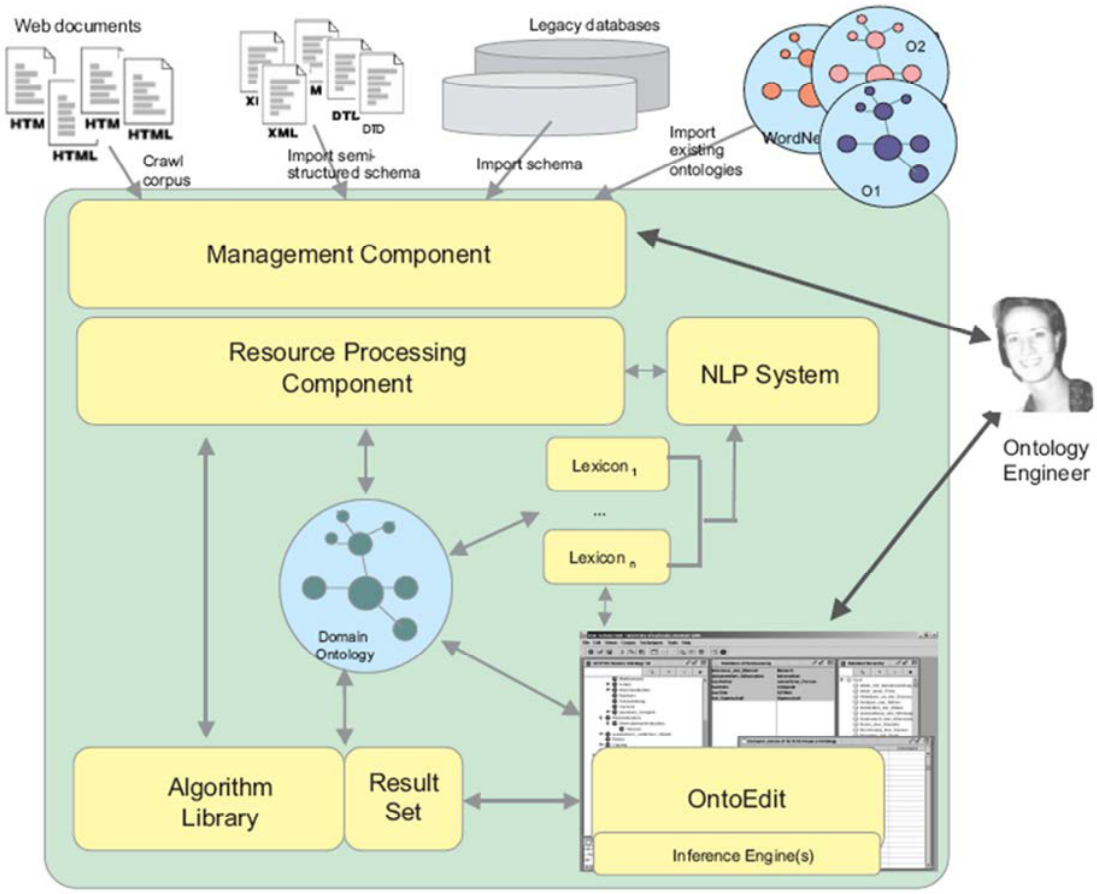


Figure 2-3: Architecture for learning ontologies for the Semantic Web (Maedche 2002)

In this architecture, the management component is used to input relevant data resources, such as HTML and XML documents, document type definitions, databases (e.g., WordNet), or existing ontologies. The resource processing component is used to transform the input data into an algorithm-specific relational representation. Based on the application domain of concepts, a suitable algorithm is selected from the algorithm library, such as association rules, clustering, and formal concept analysis, to create required results.

On the other hand, the process of ontology learning may involve six different aspects related with the structure of an ontology: term identification, synonym identification, concept identification, taxonomy construction, semantic relations extraction and rule acquisition. Three main approaches of ontology learning are the integration of ontologies,

the construction of a new ontology, and the specialization of a generic ontology (Alves et al. 2009).

For ontology extension, non-taxonomic relations that connect semantically related concepts need to be discovered in the ontology learning process (Sánchez & Moreno 2006). It is suggested that the actions (verbs) between subjects and objects are crucial to be extracted from a specific domain and to be added into the domain ontology. For example, in (Alves et al. 2009), non-taxonomic relations are identified by exploiting the verbs that represent actions or relations between concepts in sentences. Reference (Schutz & Buitelaar 2005) provides a tool for relation extraction from text in ontology extension.

In addition to ontology learning, another technology of ontology engineering has recently been developed based on the ontology bases. This technology is known as ontology mining which has opened an advanced step of Semantic Web technology by extracting ontological information or interesting concepts from existing resources, such as Web-pages, databases and other ontologies (Li & Zhong 2006; Xiaohui, Yuefeng & Nayak 2008).

2.3.8. Ontology Reasoning

When ontologies are developed in applications, ontology reasoning is an indispensable process to answer queries about the domain, and to retrieve concepts of an ontology (Horrocks & Peter 2011). Ontology reasoning facilitates drawing conclusions from existing facts by referring a set of rules which are represented in the axioms of an ontology, so that queried information can be retrieved. Axioms are able to be specified in some basic types: instantiation, assertion, subsumption, domain, range, disjointness (Grimm et al. 2011). For example, an instantiation axiom assigns an instance to a concept class, and an assertion axiom assigns two instances by means of a property.

The standards enabling reasoning on the Semantic Web are RDF Schema, the ontology language OWL, and RIF (Rule Interchange Format) (Harth, Janik & Staab 2011). Some ontology constructs, such as *rdfs:subClassOf*, *owl:sameAs*, and property chains, might be used for reasoning. The following are some tools of OWL reasoning.

- Pellet is a complete OWL-DL reasoner with extensive support for reasoning with individuals, user-defined data types, and debugging support for ontologies (http://www.mindswap.org/2003/pellet/)
- Other OWL-DL reasoners: RacerPro (http://www.racer-systems.com/) and FaCT^{++} (http://owl.man.ac.uk/factplusplus/)

2.3.9. Ontology Evaluation

It is not easy to evaluate if the ontology is valid or invalid, right or wrong, good or not good. Ontology evaluation should consider two parts: ontology verification and ontology validation (Lovrenčić & Čubrilo 2008; Vrandečic 2009). Ontology verification refers to "building the ontology correctly, that is ensuring that its definitions implement correctly the requirements". Ontology validation refers to "whether the meaning of the definitions really models the real world for which the ontology was created". According to Vrandečic (2009), some aspects of an ontology, such as vocabulary, syntax, structure, semantics, representation, context, should be considered during the design of the ontology. A degree of freedom must be evaluated for each aspect of an ontology.

There are a number of methods that have been developed for ontology evaluation. It is important to make a choice of proper ontology evaluation method relevant for the domain and task.

Considering ontology verification in the aspect of structure and representation, Tartir et al. (2005) proposed a set of metrics to analyse ontology schemata and their populations (i.e., knowledge bases). And they applied successfully the metrics to ontology evaluation using a tool OntoQA (Tartir & Arpinar 2007). Some of the metrics that are used for the evaluation of ontology construction are listed below:

a. Schema metrics

Relationship Richness (RR) (Tartir et al. 2005) is defined as

$$RR = \frac{|P|}{|SC|+|P|} \quad (2.1)$$

where, *P*: the number of relationships defined in the schema; *SC*: the sum of the number of subclasses. It reflects the diversity of relationships in the ontology.

b. Instance metrics

Class Richness (CR) (Tartir et al. 2005) is defined as

$$CR = \frac{|C'|}{|C|} \tag{2.2}$$

where, C': the number of populated classes; C: the total number of classes defined in the ontology schema. It presents how instances are distributed across classes.

Class Instance Distribution (CID) (Tartir et al. 2005) indicates how instances are spread across the classes of the schema, and is defined as

$$CID = StdDev(Inst(C_i)) \tag{2.3}$$

where, $Inst(C_i)$: the number of instances per class.

Class Connectivity (Tartir et al. 2005) indicates the centrality of a class. The connectivity of a class is defined as the total number of relationships instances of the class have with instances of other classes.

Class Importance ($Imp(C_i)$) (Tartir et al. 2005) identifies which classes of the schema are in focus to extract the instances, and is defined as

$$Imp(C_i) = \frac{Inst(C_i)}{KB(CI)} \tag{2.4}$$

where, $Inst(C_i)$: the number of instances of the class C_i; $KB(CI)$: the total number of class instances in the knowledge base.

Relationship Utilization (Tartir et al. 2005) reflects how the relationships defined for each class in the schema are being used at the instances level, and is defined as

$$RU(C_i) = \frac{IREL(C_i)}{CREL(C_i)} \tag{2.5}$$

where, $IREL(C_i)$: the number of relationships that are being used by instances I_i that belongs to C_i; $CREL(C_i)$: the number of relationships that are defined for C_i at the schema level.

Considering ontology validation in the aspect of vocabulary and representation, Chen and Chuang (2008) proposed two kinds of precision evaluation methods: (1) concept precision (CP), and (2) concept location precision (CLP) to evaluate the effect of their domain ontology, which is constructed automatically based on domain keywords extracted from the collected Web-pages. The **Concept precision** demonstrates the precision of the concepts generated by the system. The **Concept location precision** presents not only the precision of the generated concepts, but also the precision of the location in the hierarchy relations. The formulae of concept precision and concept location precision (Chen & Chuang 2008) are listed below:

$$precision(CP) = \frac{A}{A+B}, \text{ and} \tag{2.6}$$

$$precision(CLP) = \frac{C}{C+D}, \tag{2.7}$$

where A is the number of concepts that the system generates and the expert has accepted; B is the number of concepts that the system generates and the expert has not accepted; C is the number of concepts that the system generates and the expert defines whose locations are right; and D is the number of concepts that the system generates and the expert defines whose location is in error.

2.4. Recommender Systems

Recommender systems (Adomavicius & Tuzhilin 2005; Mobasher 2007b) were developed to learn Web user experience in order to model the interaction between users and items described on Web-pages and to recommend the interesting items to the users. The popularity of recommender systems is increasing with the rapid growth of the Internet since the mid-1990s. In the systems, recommended items may be Web-pages (links), articles, books or products. An intelligent recommender system will support Web users to make better decisions to rapidly reach their own target pages during a browsing session. Therefore, recommender systems become more and more important in Web-based applications, such as e-commerce, e-government, and e-services.

At the beginning, traditional recommender systems have been developed merely based on Web mining. Web recommendations mostly rely on the informally represented data patterns which are discovered from Web data, e.g., Web server log files, user profile, and Web content. In the 2000s, the advent of the Semantic Web has changed the World Wide Web. The Semantic Web offers a good basis to enrich Web mining by discovering the "semantics" in the data and make the discovered knowledge explicit. Semantic Web mining (Kolari & Joshi 2004; Stumme, Hotho & Berendt 2006) has emerged as an advanced technique which can improve the effectiveness of Web mining based on the ontology technology. Ontology has significantly contributed to semantic knowledge representation in order to semantically enhance information process in recommender systems, as known as semantic (enhanced) recommender systems.

2.4.1. Basic Types of Recommender Systems

According to Ricci, Rokach and Shapira (2011), there are six types of recommender systems that vary in terms of the used knowledge, the addressed domain, and the recommendation algorithm.

- *Content-based*: the system recommends items that are similar to the ones that the user liked in the past. The similarity of items is calculated based on their features.
- *Collaborative filtering*: in the system, an active user who is surfing the Web is suggested items that other users with similar taste liked in the past. The similarity in taste of users is calculated based on the similarity in their activity history. This technique is considered to be the most popular one in recommender systems.
- *Demographic*: the system recommends items based on the demographic profile of the user.
- *Knowledge-based*: the system recommends items based on explicit domain knowledge about how certain item features meet the user needs and references, and how the items are useful for the user.
- *Community-based*: the system recommends items based on the references of the user's friends.

- *Hybrid recommender systems*: these recommender systems combine some of the above mentioned techniques by taking the advantages of the used techniques to optimize Web recommendation.

The following discusses more about hybrid recommender systems. In fact, Web personalization is the main approach of recommender systems. It recommends things based on the individual's past behaviour. WUM is a main approach to personalization in recommender systems (Jalali et al. 2009; Mobasher 2007a). The overall process of Web personalization based on WUM consists of three phases: data preparation and transformation, pattern discovery, and recommendation (Mobasher 2007a). In the pattern discovery phase, the commonly used data mining techniques are clustering, association rule mining, sequential pattern discovery, and probabilistic modelling. Mobasher (2007a) points out that hybrid data mining models can leverage data from a variety of sources and produce more effective personalized recommendations, for example the integration of user-based model with content features or semantic knowledge or linkage structure.

Regarding personalized recommendation, a mixture of Markov models based approach is proposed to cluster users and to capture the sequential relationships inherent in user access histories (Liu, Huang & An 2007). This hybrid method achieves higher performance than the Markov models, the association rules, or clustering methods. Another hybrid method modelling user behaviour, a Latent Dirichlet Allocation (LDA) model, is proposed to present WAP for collaborative recommendation making (Xu, Zhang & Yi 2008). In this model, the associations between sessions and topics, and the associations between topics and Web-pages are weighted. By interpreting these correlations, the user navigational preference distribution over topic space is discovered. Li and Wang (2008) propose a hybrid recommender system based on item-based collaborative filtering (CF) algorithm, so called the higher-order logic recommender system, which can capture how the active user rates the similar items. In this system, a higher-order logic declarative programming language, namely Escher, was used to present user data for the computation of the higher-order logic distances between instances. Those hybrid models achieve more accurate recommendations, but vulnerable to a sufficiently large attack, i.e., an attack that contains enough profiles and ratings, (Mobasher et al. 2007). According to Mobasher, B., et al.

(2007), the user-based and item-based algorithms are vulnerable in the face of "profile injection" attacks, while the semantically enhanced CF algorithm which is a weighted hybrid recommendation algorithm is more robust and accurate than the standard algorithms. It appears that the semantically enhanced hybrid algorithms can effectively reduce the impact of profile injection attack, and have shown their robustness.

2.4.2. Semantic Recommender Systems

It has been shown that Semantic Web technology opens a positive point of view of Web development. The integration of semantic knowledge with Web mining plays an important role in the development of robust recommender systems. In particular, domain ontology is useful for clustering documents, classifying pages or searching subjects. Ontology concepts could help to enhance a Web personalization process with content semantics (Eirinaki et al. 2006). In this process, an ontology is built with the concepts extracted from the documents, so that the documents can be clustered based on the similarity measure of ontology terms. Then, usage data is integrated with the ontology in order to produce semantically enhanced navigational patterns. Subsequently, the system can make recommendations, depending on the input patterns semantically matched with the produced navigational patterns.

Using ontology, the relationships between document categories accessed by users can be formally represented in order to design effective Web mining models (Li & Zhong 2003). An ontology can be extracted from accessed documents to represent the concept models of Web user information needs (Li & Zhong 2004). In such a case, the built ontology is useful to search right data when a user makes a request. On the other hand, Ruiz-Montiel and Aldana-Montes (2009) developed domain ontologies describing both users and items in order to semantically enrich recommendations, and to increase prediction accuracy. Moreover, the ontology of the discovered patterns can be expressed in XML and updated for ontology evolution (Li & Zhong 2006).

Furthermore, with the help of ontology, Semeraro et al. (2007) has proposed a semantic recommender system to support users to plan their attendance to a scientific conference. This system analyses the meaning of papers in the proceedings by lexical ontology, learns the semantic profiles (interests) from the latest papers users wrote or read, then provides

suggestions. Experimental results show that by using ontology, the system can understand relevant papers and make more accurate recommendation.

Some other works have been undertaken to use ontology-based Web mining in hybrid recommender systems with promising results. In (Garcia et al. 2010), an ontology-based recommender system is proposed, namely Generalist Recommender System Kernel (GRSK), which can be applied to any different application domains. In this work, the basic recommendation techniques including the demographic, content-based, collaborative, and general-likes filtering are used to obtain the user preferences. Based on the user preferences, interesting items can be selected. Besides, two hybrid recommendation methods including the mixed hybrid technique and the weighted hybrid technique are used to combine the lists of items and to produce the final user recommendation list. In that, ontology is used to describe the user preferences and the items for recommendation. It is designed to be generalist so that the system can be worked with any domain ontology. This system has been successfully applied to tourism domain and movies domain.

In summary, current recommender systems which are semantically enhanced by effectively using ontology to express the semantic information of application domains can achieve right recommendations. For different purposes, the discovered knowledge can be flexibly represented by extended ontologies.

2.4.3. Web-page Recommender Systems

Along with the development of recommender systems, Web-page recommendation has become increasingly popular, and is shown as links to related stories, related books, or most viewed pages at websites. When a user browses a website, a sequence of visited Web-pages during the session (the period from starting, to exiting the browser by the user) can be generated. This sequence is organized into a Web session $S = (d_1, d_2 \ldots d_k)$, where d_i (i=[1..k]) is the page ID of the i^{th} visited Web-page by the user. The objective of a Web-page recommender system is to effectively predict the Web-page or pages that will be visited from a given Web-page of a website.

There are a number of issues in developing an effective Web-page recommender system, such as how to effectively learn from available historical data and discover useful knowledge of the domain and Web-page navigation patterns, how to model and use the discovered knowledge, and how to make effective Web-page recommendations based on the discovered knowledge. A traditional Web-page recommender system might use Web usage mining techniques, such as association rules, clustering, and sequential pattern discovery, for Web-page recommendation (Gündüz-Ögüdücü 2010). For example, tree structures, e.g. WAP-tree (Pei et al. 2000), CS-Mine (Zhou, Hui & Fong 2004), and PLWAP-Mine (Ezeife & Lu 2005), and probabilistic models, e.g. WAP-tree based probability model (Dai et al. 2004), Markov chains (Zhu, Hong & Hughes 2002), and dynamic higher-order Markov models (Borges & Levene 2005), can be used to efficiently represent Web access sequences for making Web-page recommendation. However, different methods describe usage patterns in different forms, and produce different prediction accuracies on the same dataset. Combination of different methods may result in better accuracy since their benefits can be exploited while avoiding their shortcomings. For example, Bose et al. (2007) incorporate concept hierarchies into usage mining based recommendations. This approach has achieved higher performance than the traditional approaches.

Furthermore, an effective Web-page recommender system is usually based on the content and user profile models. Web mining might be used to retrieve content from user query requests, to recognize user sessions, to identify visited Web-pages, and to mine user navigation patterns in order to form a knowledge base for on-line Web-page recommendation in Web-page recommender systems, e.g., (Dixit & Gadge 2010). A hybrid approach integrating the Web-page clustering into WUM and personalization processes also achieves high performance of recommendation, as in the hybrid recommender system of Liang and Zhao (2009). In this approach, concepts are extracted from Web-pages and are used for clustering Web-pages. These clustered pages are combined with FWAP discovered from WUM in order to generate Web-page recommendations given a current WAS. Alternatively, Khribi, Jemni and Nasraoui (2009) use a collaborative modelling approach, i.e. building the user and content models, to build automatic Web-page recommendations in

e-learning platforms. In this context, the content-based filtering and collaborative filtering approaches are applied to predict a recommendation list for the active learner.

Besides, semantic-enhanced Web-page recommender systems are more powerful with the help of ontology. Integration of domain ontology with recommender systems can enrich the semantics of Web usage data to make valuable recommendations and produce promising results. In particular, ontology is used to represent an application domain and is used to express the meaning of Web-pages, so that the semantic information is able to be effectively integrated into the Web-page recommendation process (Gündüz-Ögüdücü 2010). For example, Rios and Velasquez (2008) use a concept-based approach to add semantics into the Web usage mining process; and Mabroukeh and Ezeife (2009) use semantic information to improve selective Markov models for Web prefetching. Moreover, Wei and Lei (2009) construct a website's ontology using the concepts and significant terms extracted from documents, and online recommendations are generated by semantically matching and searching frequent pages discovered from the Web usage mining process. This approach achieves the effectiveness of the precision rate, coverage rate and matching rate. On the other hand, ontology is able to be used to map semantic information to Web-pages in order to represent frequent navigational patterns extracted from WUM in the form of ontology instances, and to reason for Web-page recommendation more accurate (Salin & Senkul 2009).

In summary, it has been found that semantic-enhanced Web-page recommender systems can produce improved recommendation results. However, Web-page recommender systems have not been extensively and deeply explored in semantic Web mining.

2.4.4. Recommender System Evaluation

Regarding evaluation of recommender systems, there are some evaluation metrics designed for off-line experiments, user studies, and on-line experiments (Shani & Gunawardana 2011). It is often easiest to perform off-line experiments using existing datasets and modelling user behaviour to estimate recommendation performance measures, e.g. prediction accuracy. It is more expensive to conduct user studies by asking a small group of users about their experience of using the system through pre-defined questionnaire. On-line

experiments evaluate the system through real users' behaviours and their response to the system.

Off-line experiments are more attractive to evaluate the effectiveness of a recommender system because they require no interaction with real users, and thus allow us to compare a wide range of algorithms at low cost. Therefore, most of work for evaluating recommender systems has focused on accuracy in Web-page prediction (Gündüz-Ögüdücü 2010) in off-line experiments. In off-line experiments, the entire dataset is divided into two parts: training set and testing set. Training set is used to build the knowledge model of recommendation. Testing set is used to examine the recommendation accuracy of the built model.

The accuracy is calculated based on how well the generated recommendation sets are. Assume that the model generates a recommendation set *RSet* given a set of accessed pages *ASet* in the testing set. The accuracy of this model can be evaluated in terms of *precision* and *coverage* (or *recall*) (Nasraoui et al. 2008; Wei & Lei 2009).

$$Precision = \frac{ASet \cap RSet}{RSet} \tag{2.6}$$

$$Coverage = \frac{ASet \cap RSet}{ASet} \tag{2.7}$$

Based on *precision* and *coverage* (or *recall*), another evaluation measure, called matching rate *M*, has been provided (Wei & Lei 2009), to evaluate the best performance of a recommender model.

$$M = \frac{2 \times coverage \times precision}{coverage + precision} \tag{2.8}$$

Precision and *coverage*/*recall* are also standard metrics used to evaluate the effectiveness of information retrieval (Mobasher 2007a). In addition to *precision*, Zhou (2004) presented two other measures which are *satisfaction* and *applicability* to evaluate the performance of Web-page recommendation. These precision, satisfaction, and applicability measures will be elaborated in Chapter 3.

Selecting appropriate evaluation metrics depends on which recommendation model is used. For example, the *precision*, *satisfaction* and *applicability* metrics are suitable for the performance evaluation of a recommendation model based on association rule mining or sequence mining.

2.5. Summary

This chapter has briefly reviewed the research areas related to this study, which include Web mining, ontology, and recommender systems. It presents Web mining techniques with an emphasis on WUM which is mainly exploited in Web-page recommender systems. It also presents various ontology definitions, roles and representation languages of ontology, and lists the variants of ontologies, and some examples of ontologies. Furthermore, some background of ontology engineering, such as ontology construction, ontology learning, and ontology reasoning, and ontology evaluation methods are also presented.

Finally, the issues of recommender systems involving Web mining and ontology techniques, and evaluation metrics are discussed. It has been shown that semantic Web mining based recommender systems are powerful at present, in which Web-page recommender systems have not been explored much. The next chapter will present a comparative study on advanced sequence minding techniques and select the best one for the new Web-page recommender systems.

Chapter 3.

SEQUENTIAL PATTERN MINING FOR WEB USAGE

3.1. Introduction

Web usage mining (WUM) is a useful method of exploring Web usage data to understand Web user navigation behaviours and find useful Web usage knowledge. For an e-commerce company, WUM can be used for finding perspective customers who likely make a large number of purchases, or predicting e-commerce transactions based on the observation of previous visitors. In the context of Web-page recommender systems, WUM can be used to discover Web usage knowledge to support users to make better decisions by suggesting popular Web-pages to the users or a more efficient way to organize websites for Web-based applications, such as effective marketing (Mobasher 2007a). Choosing an effective mining algorithm plays an important role in recommending the right level of information to online users.

The goal of WUM is to capture, model, and analyse the behavioural patterns and profiles of users interacting with a website (Liu, Mobasher & Nasraoui 2011). The discovered patterns are usually a set of sequences of pages that are frequently accessed by groups of users with common interests. Sequential pattern mining algorithms are appropriate for this purpose since they can take the Web access sequences (WAS) as the input and output the frequent Web access patterns (FWAP). The sequential pattern mining algorithms can be roughly classified as the Apriori-based, pattern-growth, and Web access pattern (WAP)-tree based approaches (Mabroukeh & Ezeife 2010; Zhou 2004). The WAP-tree based approach often achieves higher performance in sequential pattern discovery, especially the PLWAP-Mine (Ezeife & Lu 2005) and CS-Mine (Zhou, Hui & Fong 2004) algorithms. However, the benefits of PLWAP-Mine and CS-Mine are not sufficiently investigated in Web-page recommender systems.

This chapter firstly investigates the main sequential pattern mining algorithms to explain why the WAP-tree based algorithms outperform the Apriori-based and pattern-growth algorithms. Secondly, it implements the two newly proposed WAP-tree based algorithms which are the PLWAP-Mine and CS-Mine algorithms in a Web-page recommender system in order to compare their performance using three real world datasets in term of execution time, the number of resulting patterns, and recommendation accuracy. The experimental results show that PLWAP-Mine is more effective than CS-Mine while CS-Mine performs faster in almost all test cases. Based on these experiments, this chapter highlights some hints about how to enhance the performance when applying these two algorithms to specific Web applications. Finally, a suitable algorithm is selected for the SWRS of this research, which will be discussed in Chapter 7.

This chapter is organized as follows: Section 3.2 discusses about data sources of WUM. Section 3.3 analyses the main existing sequential pattern mining techniques for WUM and then highlights the typical features of PLWAP-Mine and CS-Mine. Section 3.4 develops a Web-page recommender system, in which a sequence mining algorithm, e.g. PLWAP-Mine and CS-Mine, can be used to discover the FWAP for generating the Web-page recommendation rules. Section 3.5 presents a set of experiments to compare the performance of the two algorithms in the context of the Web-page recommender system, and evaluates the experimental results. Section 3.6 gives some remarks of PLWAP-Mine and CS-Mine. Finally, Section 3.7 concludes this chapter.

3.2. The Web Usage Data as the Source of Mining

The primary data sources in WUM are the Web server access logs. The log data, or called Web usage data, collected automatically by the Web servers represents the fine-grained navigational behaviour of Web users (Liu, Mobasher & Nasraoui 2011). Each hit against the server, corresponding to an HTTP request, generates a single entry in the server access logs. Each log entry may contain fields identifying the client IP address, the user ID, the request time, the requested URL, HTTP status code, etc. For example, an excerpt of the Web log from the website of the University of Saskatchewan in Canada

(http://www.usask.ca/) is shown in the Figure 3-1, in which eight partial log entries are shown. The user IDs in the log entries have been removed to protect privacy.

```
cdc8g5.cdc.polimi.it - - [01/Jun/1995:00:11:03 -0600] "GET /~friesend/tolkien/rootpage.html" 200 461
freenet2.carleton.ca - - [01/Jun/1995:00:16:54 -0600] "GET /~scottp/free.html" 200 5759
red.weeg.uiowa.edu - - [01/Jun/1995:00:18:14 -0600] "GET /~friesend/tolkien/rootpage.html" 200 461
interchg.ubc.ca - - [01/Jun/1995:00:23:53 -0600] "GET /~lowey/encyclopedia/index.html" 200 2460
interchg.ubc.ca - - [01/Jun/1995:00:24:17 -0600] "GET /~lowey/encyclopedia/help.html" 200 2570
interchg.ubc.ca - - [01/Jun/1995:00:24:59 -0600] "GET /~lowey/encyclopedia/atlas.html" 200 691
interchg.ubc.ca - - [01/Jun/1995:00:25:16 -0600] "GET /~lowey/encyclopedia/m/maps.html" 200 1426
info.curtin.edu.au - - [01/Jun/1995:00:25:25 -0600] "GET /~scottp/publish.html" 200 271
```

Figure 3-1: Sample Web log from website http://www.usask.ca/

In the sample Web log (Figure 3-1), log entry 1 shows a user with IP address "cdc8g5.cdc.polimi.it" accessing Web-page: /~friesend/tolkien/rootpage.html" on the server (http://www.usask.ca/). Log entries 2 and 3 show two other users who have arrived at Web-pages "/~scottp/free.html" and "/~friesend/tolkien/rootpage.html", respectively. Log entries 4-7 show that a user with IP address "interchg.ubc.ca" has navigated from "/~lowey/encyclopedia/index.html" to access other pages: "/~lowey/encyclopedia/help.html", "/~lowey/encyclopedia/atlas.html", and "/~lowey/encyclopedia/m/maps.html". Log entry 8 shows that page "/~scottp/publish.html" is visited by another user.

According to Liu et al. (2011), depending on the goals of the analysis, the log data needs to be transformed and aggregated at different levels of abstraction. In this study, the most basic level of data abstraction is that of a **pageview**. A pageview is an aggregate representation of a collection of Web objects representing a specific "user event", e.g., reading an article, viewing a course, or reserving a book at a digital library. At the user level, the most basic level of behavioural abstraction is that of a **session**. A session is a sequence of pageviews by single user during a single visit.

An important task in the acquisition of useful knowledge is the creation of a suitable target dataset to which applications. The data preparation process is often the most time consuming and computationally intensive step in the Web usage mining process, and often requires the use of special tools. This process may involve pre-processing the original data,

and transforming the processed data into a form suitable for input into specific data mining operations. The log data contains a huge amount of historical information about Web usage activities of users who visited the website. In order to acquire usable data, it is necessary to clean Web logs.

Web log cleaning (Liu, Mobasher & Nasraoui 2011) is a basic technique which is often performed in the pre-processing process before building a knowledge base in a formal format. It involves tasks such as, removing extraneous references to embedded objects, e.g., style files, image files, or sound files. The cleaning process may also involve the removal of at least some of the data fields (e.g., number of bytes transferred or version of HTTP protocol used, etc.) that may not be useful in data analysis or mining tasks. A Web log cleaning tool[7] is used to create datasets of Web sessions. Each session is a WAS requested by a user. These WAS will be the input data of Web usage mining algorithms.

3.3. Sequential Pattern Mining

WUM techniques address the problem of discovering users' Web usage behaviours from their Web access activities. Web usage data is of sequential nature, i.e., each piece of data is an ordered list/sequence of events, e.g. visited Web-pages. Hence, the main WUM techniques are sequential pattern mining techniques. Therefore, this section analyses the main existing sequential pattern mining techniques as WUM techniques, highlights two outstanding algorithms, i.e., PLWAP-Mine and CS-Mine, and then provides the performance comparisons of these two sequential pattern mining algorithms.

3.3.1. Sequential Pattern Mining Algorithms

Sequence mining algorithms focus on discovering useful patterns in sequential datasets. Real world sequential datasets include e-commerce transactions, banking transactions and customer queries. In this chapter, the dataset is Web access sequences or WAS, which are the navigational sequences of events, and the goal of sequence mining algorithms is to extract frequent Web access patterns from WAS. A frequent Web access pattern is a

[7] http://sol.cs.uwindsor.ca/~cezeife/webcleaner.tar.gz, Accessed January, 2011

sequence of the events that frequently occurred in a specific order. A sequential pattern is a subsequence of a Web access sequence. A support threshold is used to filter FWAP from WAS.

Definition 3.1 *(Support)* (Mobasher et al. 2002)

Given a set Δ of WAS, and a set $\mathbb{P} = \{P_1, P_2 \ldots P_n\}$ of frequent (contiguous) sequential patterns over Δ, the support of each $P_i \in \mathbb{P}$ is defined as follows:

$$\sigma(P_i) = \frac{|\{S \in \Delta : P_i \subseteq S\}|}{|\Delta|}, \quad (3.1)$$

where S is a WAS.

In general, sequential pattern mining algorithms can be used to extract certain sequential patterns whose supports are equal or greater than a predefined minimal support threshold (*MinSup*). The existing sequential pattern mining algorithms fall into roughly three main approaches: the Apriori-based, pattern-growth and WAP-tree based approaches. The following will analyse the main existing sequential pattern mining algorithms from these three approaches, and highlight their advantages and disadvantages.

1) Apriori-based Approach

From this approach, the AprioriAll was the first algorithm introduced for mining sequential patterns by Agrawal and Srikant (1995). This algorithm scans the WAS database several times to find frequent item-sets of size k at each k^{th}-iteration (starting from k=2). The set of candidate sequences C_k in the k^{th}-iteration are generated by joining the frequent sequences L_{k-1} in the $(k\text{-}1)^{th}$-iteration with itself, and then are pruned. The frequent sequences L_k in the k^{th}-iteration are created by adding all sequences from C_k with support ≥ *MinSup* until there are no more candidate sequences. The AprioriAll technique suffers from heavy overhead in terms of memory space and execution time because of this generate-and-test process.

Following the AprioriAll, a generalized sequential pattern (GSP) mining algorithm was proposed by Srikant and Agrawal (1996). Similar to the AprioriAll algorithm, GSP adopts a multiple-scan candidate generate-and-test method for finding sequential patterns. For each

level (i.e., sequences of length-*k*), it scans the database to collect a support count for each candidate sequence and then generates candidate length-(*k*+*1*) sequences from length-*k* frequent sequence using Apriori. This process repeats until no frequent sequence or no candidate can be found. Experimental results have shown that GSP is 20 times faster than the AprioriAll and efficient when the sequences are not too long and the database is not too large.

After the GSP algorithm, Zaki (2001) developed the sequential pattern discovery using equivalent class (SPADE) algorithm which grows the sub-sequences (patterns) one item at a time by using the Apriori candidate generation. Faster than previous methods, SPADE usually makes only three database scans, so it is more efficient in dealing with large databases.

2) Pattern-Growth Approach

After the Apriori-based approach, the pattern-growth methods emerged as a solution to the problem of generate-and-test in the early 2000s. The main idea of the approach is to avoid generating candidate sequences altogether and to only search a restricted portion of the initial database (Han, Pei & Yan 2005). The pattern-growth algorithms aim to build a representation of the database, then to partition the search space in a particular way, and to generate as few candidate sequences as possible by growing on the already mined frequent sequences, and applying the Apriori property to search for frequent sequences. The typical pattern-growth algorithms are FreeSpan (Han et al. 2000) and PrefixSpan (Pei et al. 2001), which construct projected databases which are the collections of suffixes of sequences in the original database with respect to the prefixes based on the current sequential patterns (Han, Pei & Yan 2005). The database projection allows a reduction in the search space at each recursive step in constructing patterns. Compared with FreeSpan, PrefixSpan is more efficient because the projected databases keep shrinking when prefix grows. Furthermore, PrefixSpan does not generate any candidate, and only counts the frequency of items. Because of these major costs, PrefixSpan outperforms the Apriori-based algorithms, e.g., GSP and SPADE, and FreeSpan in most of cases (Han, Pei & Yan 2005).

3) WAP-tree based Approach

The WAP-tree based approach aims to build a tree-structure of the WAS database to overcome the limitations of Apriori-like algorithms and to extract frequent patterns efficiently. At the beginning, Han et al. (2000) proposed a frequent pattern tree (FP-tree) to store the crucial information of frequent patterns. Based on the FP-tree, a large database can be compressed into a condensed and smaller data structure so that mining the complete set of frequent patterns becomes faster and more effective. More importantly, using a FP-tree avoids the problems of multiple database scan and the generation of an explosive amount of candidates. In addition, using a FP-tree converts the search for the FPs in the large search space to traverse a FP-tree so that the search becomes more efficient.

To fulfil the potential of a FP-tree, Pei et al. (2000) proposed a highly compressed data structure, namely Web access pattern tree (WAP-tree), to store the database of WAS. Constructing a WAP-tree entails scanning the database twice: the first scan is to find all frequent individual events, and the second is to construct a WAP-tree over the set of frequent individual events of each session. The WAP-tree facilitates the development of mining algorithms which can handle a large database of Web access patterns, such as WAP-Mine (Pei et al. 2000), Conditional Sequence Mining (CS-Mine) (Zhou, Hui & Fong 2004) and pre-order linked WAP-tree mining (PLWAP-Mine) (Ezeife & Lu 2005). By mining a WAP-tree for frequent patterns, these algorithms avoid the problem of generating the explosive number of candidates as encountered in the Apriori-based algorithms. Some experimental results have shown that the WAP-based algorithms perform faster than traditional sequence mining techniques (e.g., the Apriori-based algorithms). The main reason is that the WAP-based algorithms use the compact structure of WAP-tree, which is a fundamental advancement compared with the Apriori-based algorithms. Moreover, the WAP-tree allows some novel sequence search strategies effectively used in the mining process. For example, the WAP-Mine algorithm recursively constructs the intermediate conditional WAP-tree by using the conditional suffix patterns. Two outstanding algorithms from this approach are PLWAP-Mine and CS-Mine.

3.3.2. Pre-order Linked Web Access Pattern Tree Mining

The PLWAP-Mine algorithm scans the database of WAS twice to find all frequent individual events and construct a PLWAP-tree over the set of individual frequent events (Ezeife & Lu 2005). While constructing the PLWAP-tree, the binary position codes are assigned to each node of the tree. The binary code assignment technique is performed by using a rule similar to Huffman code generation (Huffman 1952). Based on the position codes, the algorithm can determine the suffix trees of any prefix event of frequent patterns. Therefore, the algorithm can recursively mine the PLWAP-tree using common prefix pattern search to find out all FWAP. Frequent *m*-sequences are computed and discovered using frequent *(m-1)*-sequences and the appropriate suffix sub-trees. As a result, a complete set of frequent patterns are efficiently discovered from the search space, i.e., the PLWAP-tree.

3.3.3. Conditional Sequence Mining

The CS-Mine algorithm scans the database of WAS once to build a WAP-tree. Given the WAP-tree, the conditional sequence base of each frequent event is initialized. That means the database is sub-divided to search frequent patterns based on these conditional sequence bases. If the combination of the conditional sequences of each frequent event is a single sequence, then a frequent pattern will be generated. If this combination is failed, then the sub-conditional sequence base of each frequent event will be re-constructed and the algorithm recursively test if a frequent pattern exists. Two major processes in this algorithm are database division and the combination of conditional sequences. The database division makes search space smaller than other sequence mining algorithms, while the combination of conditional sequences can obtain a set of frequent patterns in a limited search space.

As a result, the algorithm might miss out some frequent patterns because it does not consider the combination of all sequences in the entire search space. Although experimental results show that CS-Mine algorithm is significantly fast in cases of smaller support threshold and larger database size, compared with other sequence mining algorithms, the effectiveness of the frequent WAP obtained from this method is questionable for making user behaviour predictions.

3.3.4. Comparison

This sub-section compares the abovementioned sequential pattern mining techniques based on Mabroukeh and Ezeife's (2010) works and other studies (Ezeife & Lu 2005; Han, Pei & Yan 2005; Srikant & Agrawal 1996) as follows.

- The main drawback of most of the Apriori-based algorithms is that the bottlenecks of candidate generating-and-testing may occur when applied to mine long sequential patterns due to the huge set of candidates that need to be generated and the need of multiple scans of database. In other words, most of these algorithms have the limitations of costly candidate generation and multiple database scan, so their performance is often not satisfactory in mining long patterns.
- The algorithms from the pattern-growth approach perform faster than the Apriori-based algorithms except for SPADE which is faster than FreeSpan. Generally speaking, they are more efficient in dealing with large sequence databases because they avoid the expensive candidate generate-and-test and reduce the times of database scan.
- Overcoming the problems of candidate generation-and-test and multiple database scan, the WAP-tree based approach aims to build a WAP-tree, an effective solution of data storage and retrieval. Based on the WAP-tree, sequence mining strategies can be applied more efficiently. As explained earlier, WAP-Mine encounters the costly reconstruction of intermediate WAP-trees during mining, while CS-Mine and PLWAP-Mine avoid this problem (Ezeife & Lu 2005; Zhou, Hui & Fong 2004).

The performance comparisons of the sequential pattern mining algorithms from the abovementioned three approaches are summarized in Table 3-1, in which the algorithms are sorted in descending order of the execution time.

It can be seen that PrefixSpan is more efficient than the Apriori-based algorithms because it avoids the generation-and-test and multiple scans of the database. However, PrefixSpan cannot compete with WAP-Mine, CS-Mine and PLWAP-Mine because they use an uncompressed data structure. In other words, the tree-based algorithms outperform

the other sequence mining techniques. Among the WAP-tree based algorithms, PLWAP-Mine and CS-Mine outperform WAP-Mine.

Table 3-1: The performance of sequential pattern mining algorithms

Algorithm	References	Candidate generation-and-test	Multiple scans of the database	Compression
AprioriAll *Apriori-based*	Agrawal and Srikant (1995)	Yes	Yes	No
GSP *Apriori-based*	Srikant and Agrawal (1996)	Yes	Yes	No
FreeSpan *Pattern-growth*	Han et al. (2000)	No	No	No
SPADE *Apriori-based*	Zaki (2001)	Yes	Yes	No
PrefixSpan *Pattern-growth*	Pei et al. (2001)	No	No	No
WAP-Mine *WAP-tree*	Pei et al. (2000)	No	No	Yes
PLWAP-Mine *WAP-tree*	Ezeife and Lu (2005)	No	No	Yes
CS-Mine *WAP-tree*	Zhou et al. (2004)	No	No	Yes

3.4. Web-page Recommendation using Tree-based Mining Techniques

Mining Web access sequences effectively for Web-page recommender systems has been a big challenge due to the large number of different items in the WAS and the highly variable lengths of these sequences. A sequential pattern mining algorithm plays a crucial role in Web-page recommender systems. From the review in the previous section, it has been found that PLWAP-Mine and CS-Mine are heading the other sequential pattern mining algorithms. This section will present an experimental study of these two algorithms in the context of a Web-page recommender system.

3.4.1. Web-page Recommender System Architecture

As part of the SWRS of this research, a Web-page recommender system is developed purely based on WUM. The architecture of the Web-page recommender system is shown in

Figure 3-2. This architecture consists of two parts: the off-line mining; and the on-line recommendation.

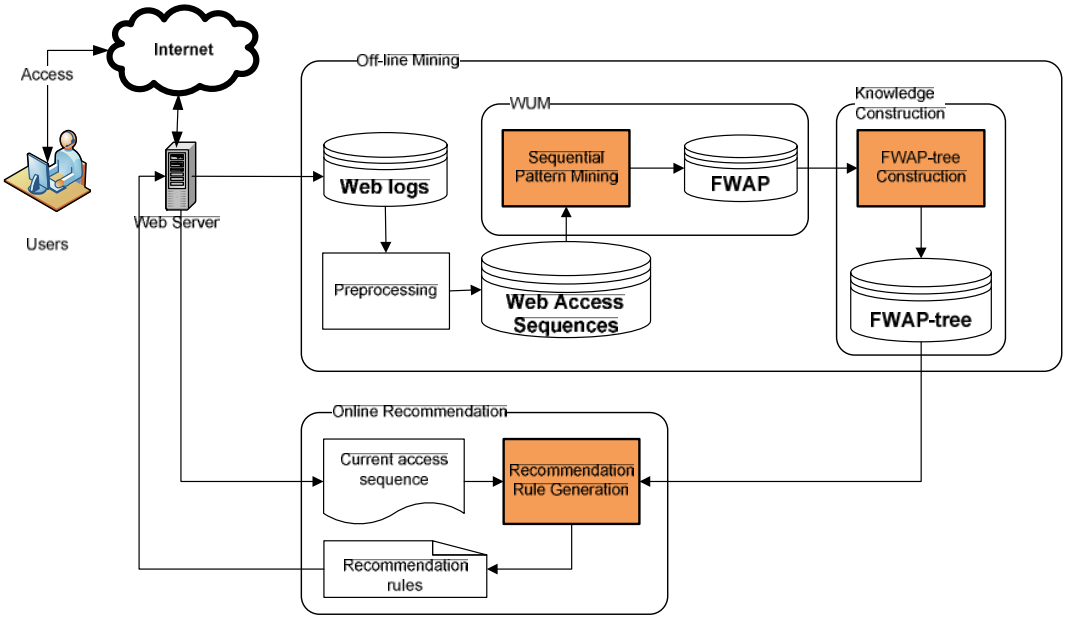


Figure 3-2: Web-page recommender system architecture

1) Off-line mining

As explained earlier, the input to this part is the Web log obtained from a Web server of the given website, which is a record set of all users' Web navigation history (i.e., WAS). Each event in WAS is a Web-page accessed by a user.

The *pre-processing* component filters out the unnecessary information in the Web access sessions to produce a dataset of WAS.

The *sequential pattern mining* component mines the WAS, and generate a complete set of FWAP by using the PLWAP-Mine or CS-Mine algorithms (referring to Section 3.4.2). Each FWAP is a sequential pattern of frequently visited Web-pages at the given website, and its support is equal or greater than a pre-defined minimum support threshold (*MinSup*).

The *FWAP-tree construction* component constructs a FWAP-tree which is a compact data model for representing the discovered FWAP (referring to Section 3.4.3), and to facilitate the generation of recommendation rules in the online recommendation phase.

2) On-line recommendation

When a user visits some pages of the website, the user's HTTP requests are recorded in a sequential order. The sequence of these requests is processed to construct a current Web access sequence which will be fed into this online recommendation part.

The *recommendation rule generation* component takes the user's current Web access sequence and generates a set of recommendation rules by matching this Web access sequence with the frequent Web access patterns stored in the FWAP-tree. Using a recommendation rule, if the user has accessed the matched Web-pages (the left hand side of the rule) then we can predict that he or she would visit the next pages predicted in the FWAP-tree (the right hand side of the rule). Based on the recommendation rules, the interesting Web-page links will be recommended for the user's next move/step in his or her navigation.

The following sub-sections will present the sequential pattern mining, FWAP-tree construction, and recommendation rule generation components in details.

3.4.2. Sequential Pattern Mining Component

The two chosen algorithms, i.e. PLWAP_Mine and CS-Mine, are implemented in this component.

Let D be a set of accessed Web-pages. A Web access sequence can be represented as $S = d_1 d_2 \ldots d_l$ ($d_i \in D$ for $1 \le i \le l$), l is the length of S. The algorithms, as Algorithms 3-1 and 3-2, are listed below.

Algorithm 3-1: PLWAP-Mine

Input: Web access sequence database (WASD), minimum support *MinSup* ($0 < MinSup \leq 1$)
Output: complete set of frequent patterns in WASD
Process:
Scan WASD once, find all frequent individual events.
Scan WASD again to obtain the frequent sequence in each WAS, construct a PLWAP-tree over the set of individual frequent events by inserting the frequent sequences. Each node in the tree registers three pieces of information: node label, node count, and node position code.
Recursively mine the PLWAP-tree using common prefix pattern search algorithm.

Source: Ezeife and Lu (2005)

Algorithm 3-2: CS-Mine

Input:
MinSup (support threshold)
Γ_{WAP} (WAP-tree built from a WASD with *MinSup*)
$D = \{d_i : 1 \leq i \leq n\}$ (all accessed pages) in Header Table of WAP-tree *T*)
Output: $\mathcal{P}$ (set of frequent Web access patterns)
Process:
1: Initialize the set of Web access patterns WAP = ∅.
2: For each event $d_i \in D$ do
a. Set Conditional Suffix $S_c = d_i$, and construct the initial conditional sequence base of S_c by following d_i – queue in WAP-tree Γ_{WAP}.
b. Construct event queues for the conditional sequence base of S_c, denoted as CSB(S_c).
c. Test single sequence for CSB(S_c).
d. If test is successful, insert all ordered combinations of items in frequent sequence FS = *SingleSeq* + S_c into $\mathcal{P}$.
e. Otherwise, for each d_j in Header Table of CSB(S_c), construct sub-conditional sequence base of d_j of CSB(S_c), set $S_c = d_j + S_c$. Recursively mine CSB(S_c) from step 2(b).
3: Return $\mathcal{P}$.

Source: Zhou et al. (2004)

3.4.3. Frequent Web Access Pattern Tree Construction Component

According to Zhou (2004), a new FWAP-tree construction algorithm is developed to better support the generation of recommendation rules. The detailed FWAP-tree construction algorithm is shown in Algorithm 3-3. In this algorithm, we only scan the set of FWAP once to construct a FWAP-tree. Unlike (2004), during the scan, nodes will be created and labelled with a symbol (of each item) and a corresponding support value, which counts the number of hits on the item.

Algorithm 3-3: FWAP-tree construction

Input: $\mathcal{P}$ (set of FWAP)
Output: Γ_{FWAP} (FWAP-tree)
Process:
Create an empty root node R for FWAP-tree Γ_{FWAP}
For each pattern $P \subset \mathcal{P}$, denoted as $P = d_1\ d_2\ \ldots d_n$ ($d_i \in D$ for $1 \le i \le n$), do
Set *current_node* point to R
For i = 1 to n do
If *current_node* has a child labelled d_i {
Set the support of d_i increase 1,
Set *current_node* point to d_i
} Else {
Create a new child node, labelled d_i with support = 1
Set *current_node* point to the new child node
}
Return FWAP-tree Γ_{FWAP}

3.4.4. Recommendation Rule Generation Component

To predict the next Web-pages to be accessed by the given user, we need to match the user's current WAS with the FWAP stored in the FWAP-tree. The suffix sequences of the current WAS will be considered to generate a rule, which is, if there exists a path from FWAP-tree matching with the longest suffix sequences then the next child items of the path will be recommended (Zhou 2004). The details of the algorithm are presented in Algorithm 3-4.

Algorithm 3-4: Recommendation rule generation

Input:
Γ_{FWAP} (FWAP-tree based on a support threshold *MinSup*)
$S = d_1\ d_2\ \ldots d_n$ (current access sequence of a user)
MinLength (the minimum length of WAS)
MaxLength (the maximum length of WAS (which should be less than the depth of the FWAP-tree))
Output:
RR (recommendation rules for *S*)
Process:
1: Initialize RR=null
2: If $|S|$>*MaxLength*, then remove the first ($|S|$ - *MaxLength* + 1) items from *S*
3: If $|S|$<*MinLength*, then return *RR*, else set *current_node* point to the root of Γ_{FWAP}
4: For each item d_i ($1 \le i \le n$) of *S*:
If *current_node* has a child labelled d_i, then set *current_node* point to this child node
Else remove the first item from *S* and repeat from step 3
5: If *current_node* has child nodes, then insert these child nodes into *RR* ordered by their supports
6: Return *RR*

Source: Zhou (2004)

3.5. Experiments with the Two Chosen Sequential Pattern Mining Algorithms

With the aim of comparing the performance of the two chosen sequential pattern mining algorithms in the context of the Web-page recommender system, the following sub-sections introduce an evaluation method, and then set up experiments.

3.5.1. Evaluation of Sequential Pattern Mining Algorithms

As mentioned before, the output of the two chosen algorithms (PLWAP-Mine and CS-Mine) is used to construct FWAP-trees for Web recommendation rule generation. According to Zhou (2004), the accuracy of Web-page recommendation can be measured by precision, satisfaction and applicability. These three measures are used to evaluate the performance of the Web-page recommender system using the two algorithms, respectively.

Definition 3.2 (*Web-page recommendation rules*)

Let $S = a_1a_2 \dots a_k\, a_{k+1} \dots a_n$ be a testing Web access sequence, where $a_i \in D$ $(1 \le i \le n)$. The FWAP-tree stores $\mathcal{P} = \{P_1, P_2 \dots P_m\}$ being a set of frequent patterns discovered from the chosen sequence pattern mining algorithms. For each prefix sequence $S_{prefix} = a_1a_2 \dots a_k$ $(MinLength \le k \le n\text{-}1)$, we generate a recommendation rule $RR = \{d_1, d_2 \dots d_M\}$ using the FWAP-tree, where all pages d_i $(i=[1..M])$ are ordered by their supports, and determine the rule is a correct rule, and/or a satisfied rule, or an empty rule based on the following conditions:

- If $a_{k+1} \in RR$, RR is correct.
- If $\exists a_i \in RR$ $(k+1 \le i \le n)$, RR is satisfied.
- If $M = 0$, RR is empty.

$R = \{RR_1, RR_2 \dots RR_N\}$ be a set of recommendation rules with RR_i $(1 \le i \le N)$ being a recommendation rule, and $|R| = N$ is the total number of recommendation rules in R including empty rules.

Definition 3.3 *(Precision)* Let R_c be the sub-set of R, which consists of all *correct* recommendation rules. The Web-page recommendation *precision* is defined as:

$$precision = \frac{|R_c|}{|R|}. \quad (3.2)$$

Definition 3.4 *(Satisfaction)* Let R_s be the sub-set of R, which consists of all *satisfied* recommendation rules. The *satisfaction* for Web-page recommendation is defined as:

$$satisfaction = \frac{|R_s|}{|R|}. \quad (3.3)$$

Definition 3.5 *(Applicability)* Let R_a be the subset of R, which consists of all nonempty recommendation rules. The *applicability* for Web-page recommendation is defined as:

$$applicability = \frac{|R_a|}{|R|}. \quad (3.4)$$

The procedure of calculating these accuracy measures, referred to as Algorithm 3-5, are described below.

Algorithm 3-5: Performance evaluation

Input:
- Γ_{FWAP} (FWAP-tree)
- WAS
- *MinLength*: the minimum length of WAS
- *MaxLength*: the depth of the FWAP-tree

Output:
- Precision
- Satisfaction
- Applicability

Process:
- For each sequence S_i in WAS:
  - $S_i = a_1\ a_2\ \ldots\ a_k\ a_{k+1}\ \ldots\ a_n$
  - Remove infrequent items from the sequence S_i
  - For each $k \geq MinLength$ to (n-1):
    - $subS = a_1\ a_2\ \ldots\ a_k$
    - Given the parameters (Γ_{FWAP}, *subS, MinLength, MaxLength*), generate recommendation rules $RR=\{d_1, d_2, \ldots, d_m\}$
    - If $RR \neq null$, then increase the number of nonempty recommendation rules $|R_a|$ by 1
    - If $a_{k+1} \in RR$, then increase the number of correct recommendation rules $|R_c|$ by 1
    - If $\exists a_i \in RR\ (k+1 \leq i \leq n)$, then increase the number of satisfied recommendation rules $|R_s|$ by 1
    - Increase the number of recommendation rules $|R|$

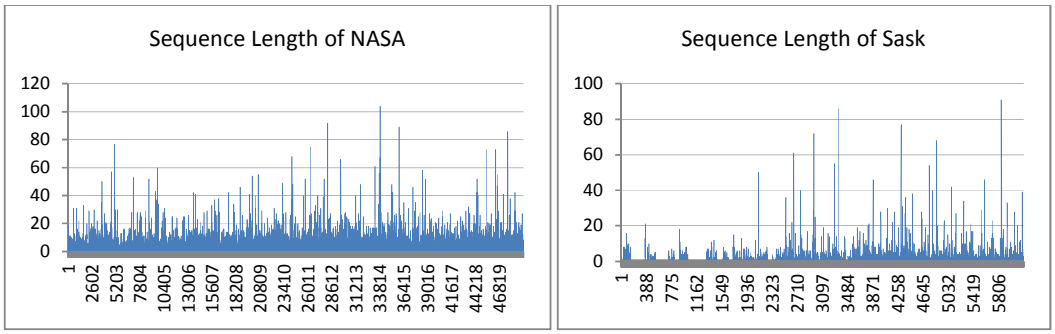
Return: the precision, satisfaction, applicability of Γ_{FWAP}

3.5.2. Training and Testing Datasets

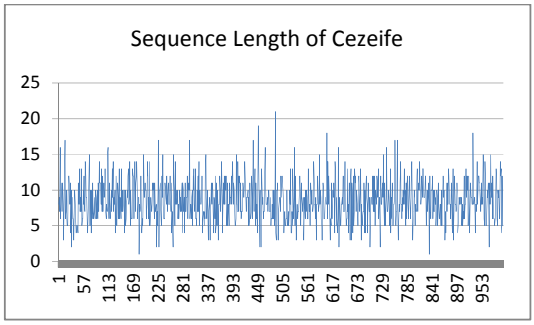
Three Web log files obtained from three real world websites are selected for experiments. The first two log files are obtained from the NASA Kennedy Space Center (NASA dataset) and the University of Saskatchewan (Sask dataset), respectively, and can be freely downloaded at http://ita.ee.lbl.gov/html/traces.html. These two Web logs were created in 1995, and have been commonly used in many research works. The third log file (Cezeife) was generated from the Web log data from the School of Computer Science, University of Windsor, and is downloadable from http://www.cs.uwindsor.ca/~cezeife/. Some statistics of these three datasets are shown in Table 3-2. Sequence lengths of each datasets are shown in Figure 3-3.

Table 3-2: Statistic of the three real world Web access sequence datasets

Dataset	Number of sessions	Number of Unique URLs	Source
NASA	49406	1446	NASA Kennedy Space Center
Sask	6186	1745	University of Saskatchewan
Cezeife	1000	92	School of Computer Science, University of Windsor



(a) (b)



(c)

Figure 3-3: Sequence length of (a) NASA, (b) Sask, and (c) Cezeife

These three datasets are selected because they are obtained from diversified real world environments. A large percentage of NASA and Sask datasets has only one element, while there are a few records with very long sequences (e.g. about from 50 to 100 items in a sequence). The average length of sequences in NASA dataset is about 20 as shown in Figure 3-3 (a), while the sequence length of Sask varies irregularly and is mostly short, Figure 3-3 (b). In contrast, most of the records in Cezeife dataset are long sequences (e.g. about from 10 to 20 items in a sequence). The sequence length of Cezeife dataset fluctuates regularly by about 10 (Figure 3-3 (c)).

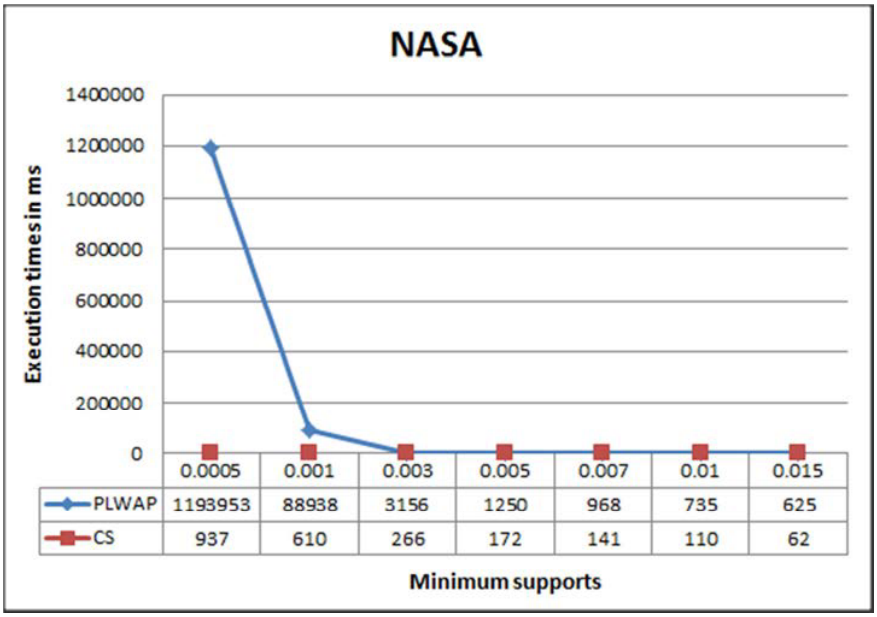
Each dataset is divided into two parts of 80% and 20% to be used for the training dataset to build the FWAP-tree and the testing dataset, respectively. The *MinLength* parameter of testing WAS is set to 2 in the following experimental cases.

3.5.3. Experimental Results and Analysis

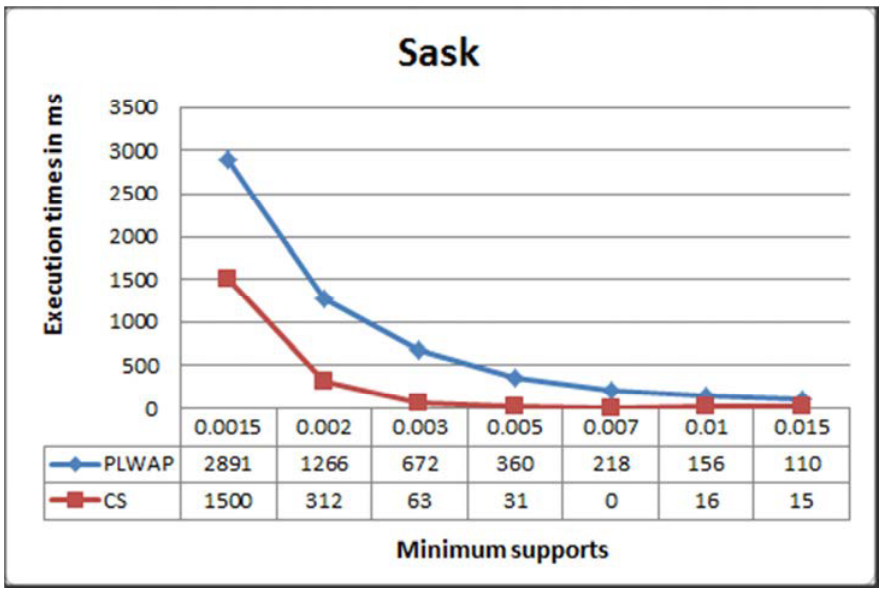
The experiments were conducted on an Intel Pentium-III Xeon processor with a CPU clock rate of 3 GHz, 3GB of main memory and running on an XP Windows platform. The mining algorithms are implemented in Java. In the experiments on the datasets outlined in the previous sub-section, the execution time and the number of frequent patterns generated by the two chosen mining algorithms were calculated. The precision, satisfaction and applicability of the Web-page recommendations based on the two chosen algorithms were also calculated against the three datasets using different minimum supports. Comparison results of the experiments are shown in the following.

1) Comparison 1: Execution times with different supports

Figure 3-4 shows the execution times of the two mining algorithms with different minimum support threshold values (from ~0.2% to ~1.5%) against the three datasets. It can be seen from Figure 3-4 that the run time of PLWAP-Mine increases sharply as the support threshold is reduced to less than 0.3%, while the run time of CS-Mine is still steady, except for the case of Sask. This shows that there is a distinct divergence between PLWAP-Mine and CS-Mine when the support threshold becomes small. CS-Mine performs less effectively when the length of sequences is not steady, as in the case of Sask. In other words, CS-Mine is sensitive to the average length of sequences.

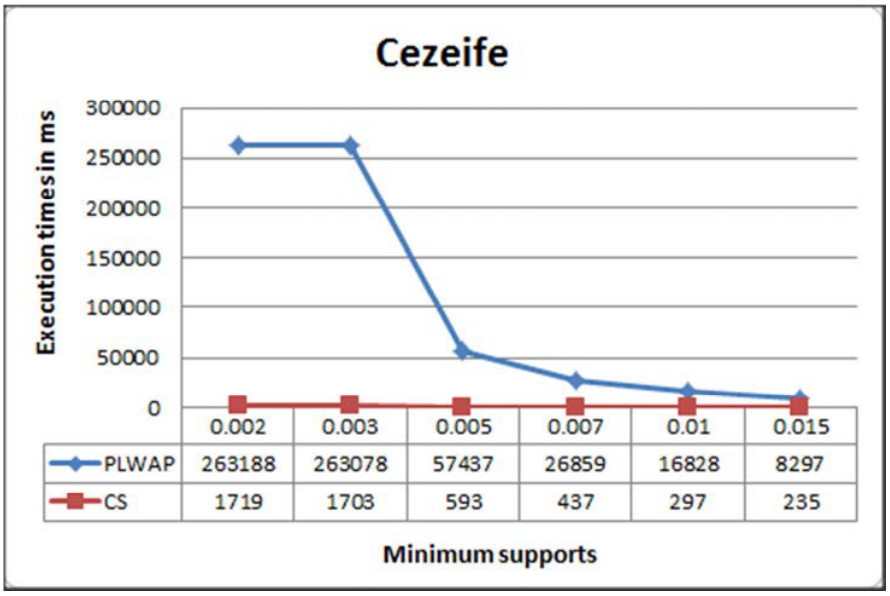


(a)



	0.0015	0.002	0.003	0.005	0.007	0.01	0.015
PLWAP	2891	1266	672	360	218	156	110
CS	1500	312	63	31	0	16	15

(b)

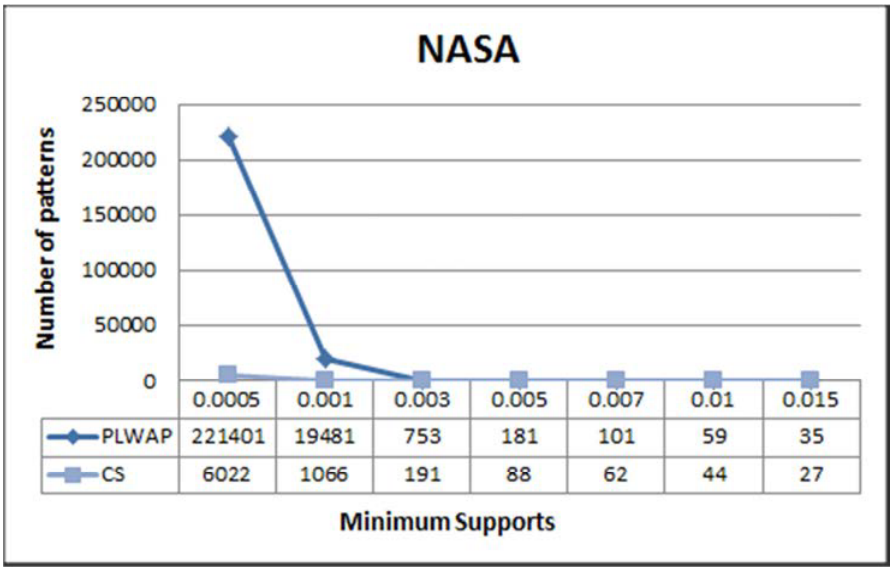


	0.002	0.003	0.005	0.007	0.01	0.015
PLWAP	263188	263078	57437	26859	16828	8297
CS	1719	1703	593	437	297	235

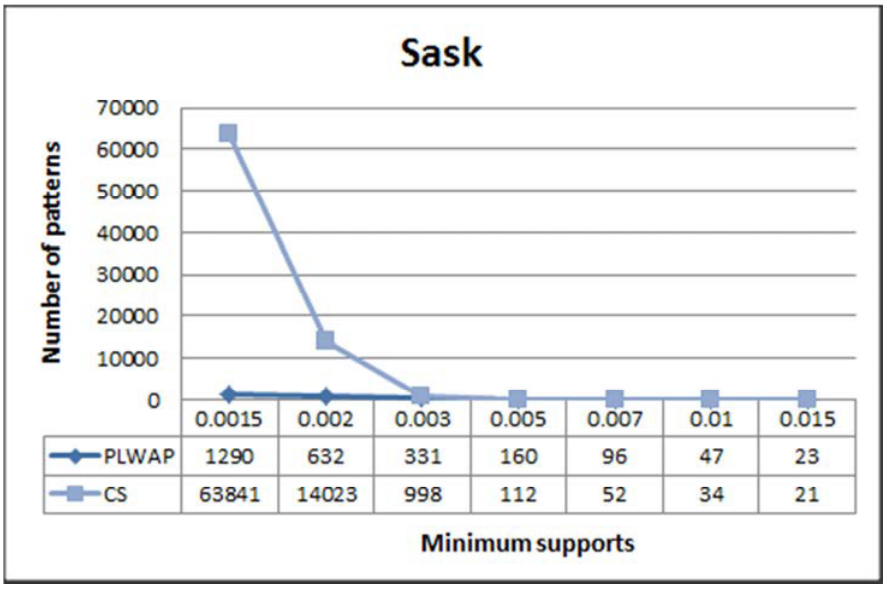
(c)

Figure 3-4: Execution times of the two mining algorithms with different supports

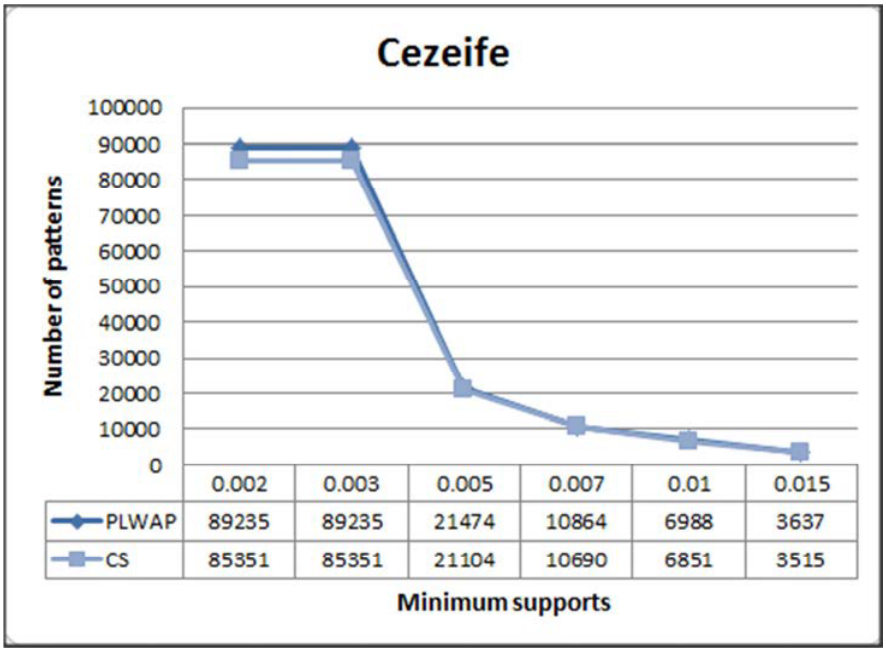
2) *Comparison 2: The number of frequent patterns with different supports*



(a)



(b)



(c)

Figure 3-5: The number of frequent patterns obtained from the two mining algorithms with different supports

Figure 3-5 shows the number of frequent patterns obtained from the two mining algorithms with different minimum support threshold values (from ~0.2% to ~1.5%) against the three datasets. It has been observed in this experimental case that the sets of patterns generated by CS-Mine are subsets of the patterns generated by PLWAP-Mine in the most test cases. This might explain why the execution time of CS-Mine is always much less than that of PLWAP-Mine as shown in Figure 3-4.

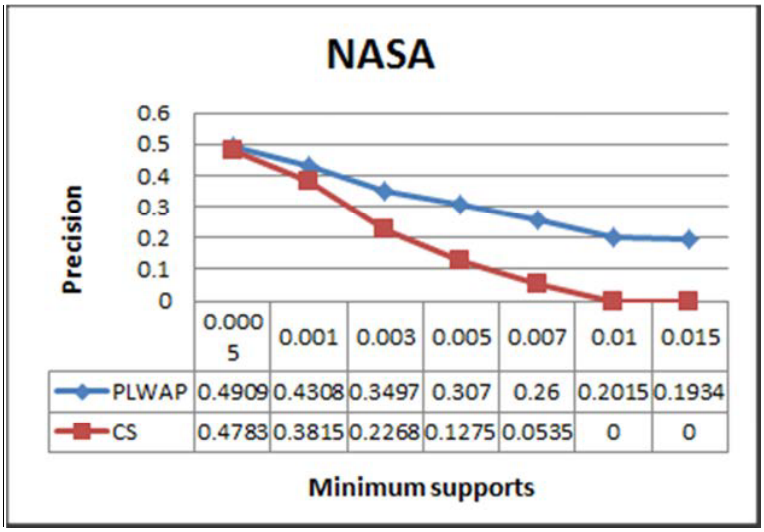
3) Comparison 3: Performance of Web-page recommendation based on PLWAP-Mine and CS-Mine

Figure 3-6 shows the precision, satisfaction and applicability of recommendation rule generation based on the two chosen algorithms in each test case. Based on the results shown in Figure 3-6, we have the following observations:

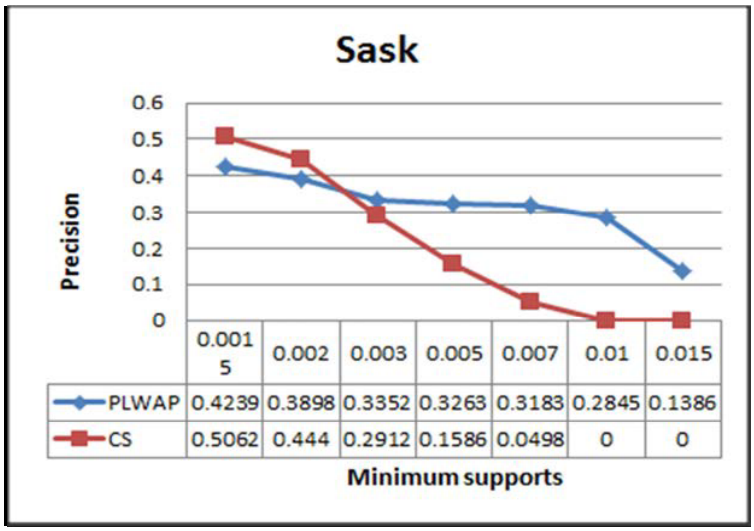
- The performance of Web-page recommendation based on PLWAP-Mine is better than CS-Mine in general, but these two algorithms show very similar performance

when the minimum support threshold values are small enough (a support of 0.05% in NASA and a support of 0.3% in Sask).

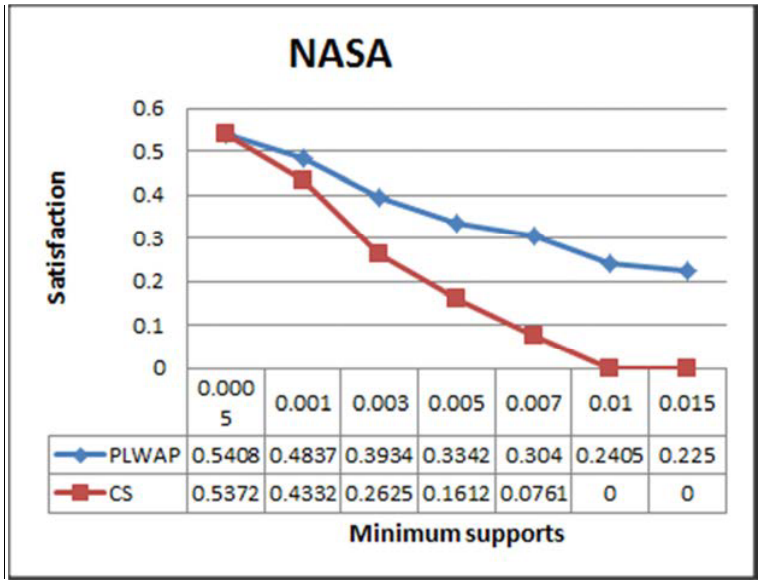
- CS-Mine performs more efficiently when the support threshold value becomes small (less than 0.3%), as in the case of Sask.
- The accuracy of CS-Mine is similar to the one of PLWAP-Mine when these two algorithms are run against the Cezeife dataset as seen from the diagrams [Figures 3-6(j), 3-6(h), and 3-6(i)].



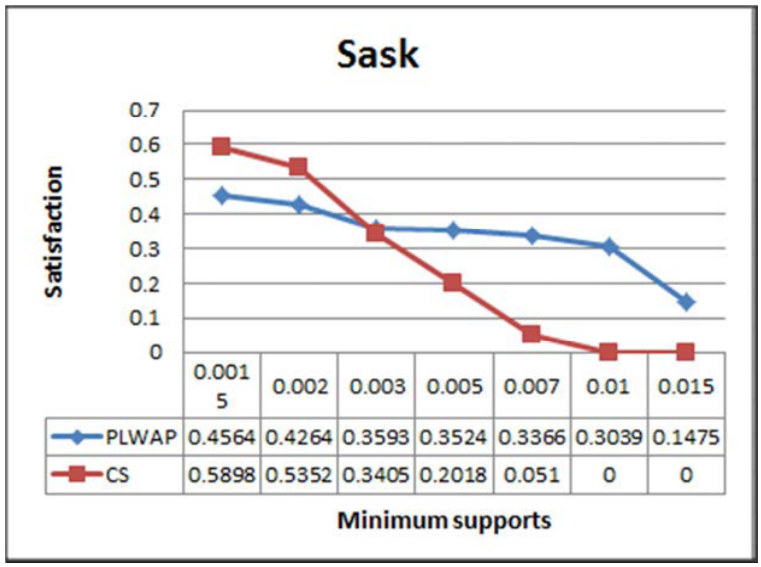
(a)



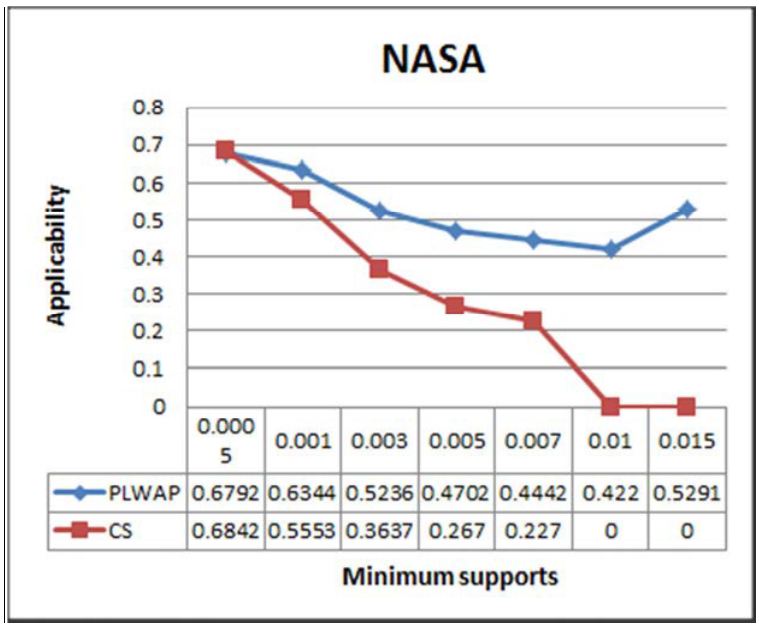
(b)


NASA
Satisfaction
0.6
0.5
0.4
0.3
0.2
0.1
0
0.0005 0.001 0.003 0.005 0.007 0.01 0.015
PLWAP 0.5408 0.4837 0.3934 0.3342 0.304 0.2405 0.225
CS 0.5372 0.4332 0.2625 0.1612 0.0761 0 0
Minimum supports

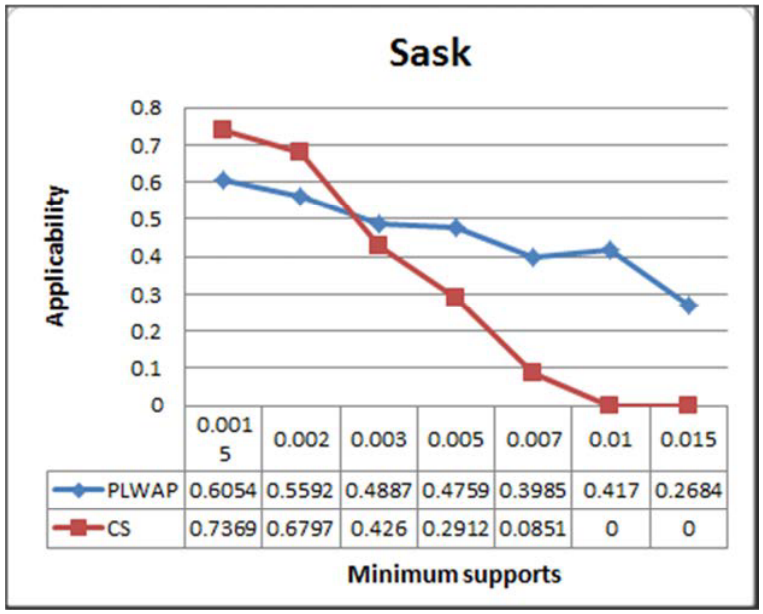
(c)


Sask
Satisfaction
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.0015 0.002 0.003 0.005 0.007 0.01 0.015
PLWAP 0.4564 0.4264 0.3593 0.3524 0.3366 0.3039 0.1475
CS 0.5898 0.5352 0.3405 0.2018 0.051 0 0
Minimum supports

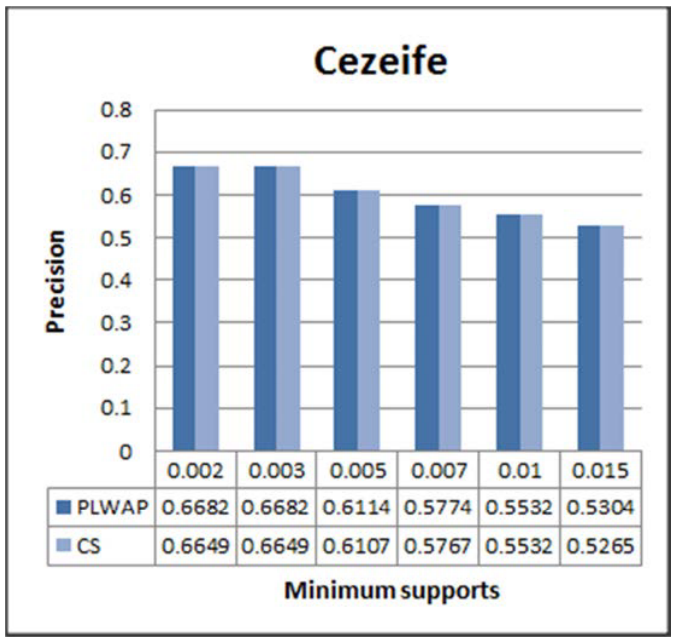
(d)


NASA
Applicability
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.0005 0.001 0.003 0.005 0.007 0.01 0.015
PLWAP 0.6792 0.6344 0.5236 0.4702 0.4442 0.422 0.5291
CS 0.6842 0.5553 0.3637 0.267 0.227 0 0
Minimum supports

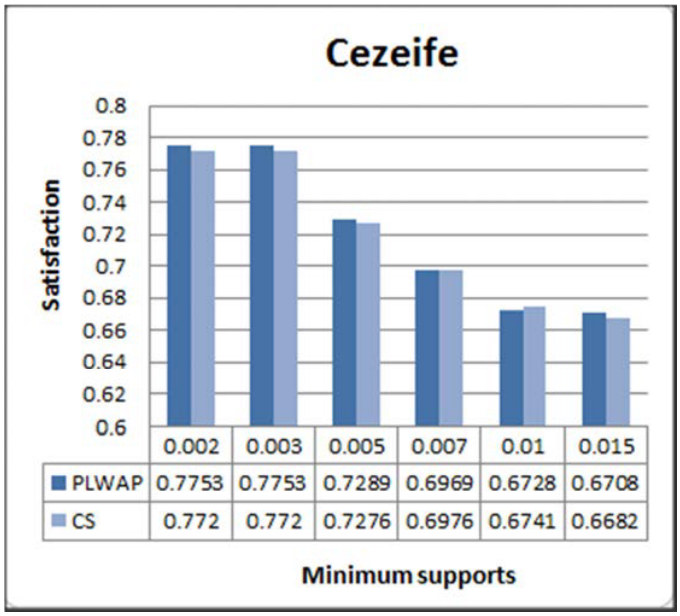
(e)


Sask
Applicability
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.0015 0.002 0.003 0.005 0.007 0.01 0.015
PLWAP 0.6054 0.5592 0.4887 0.4759 0.3985 0.417 0.2684
CS 0.7369 0.6797 0.426 0.2912 0.0851 0 0
Minimum supports

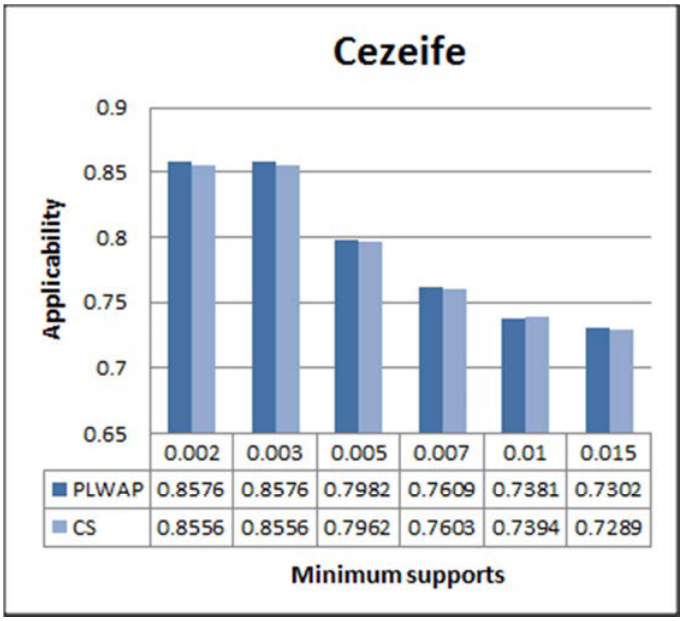
(f)


Cezeife
Precision
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.002 0.003 0.005 0.007 0.01 0.015
PLWAP 0.6682 0.6682 0.6114 0.5774 0.5532 0.5304
CS 0.6649 0.6649 0.6107 0.5767 0.5532 0.5265
Minimum supports

(j)


Cezeife
Satisfaction
0.8
0.78
0.76
0.74
0.72
0.7
0.68
0.66
0.64
0.62
0.6
0.002 0.003 0.005 0.007 0.01 0.015
PLWAP 0.7753 0.7753 0.7289 0.6969 0.6728 0.6708
CS 0.772 0.772 0.7276 0.6976 0.6741 0.6682
Minimum supports

(h)



(i)

Figure 3-6: Performance of Web-page recommendation based on PLWAP-Mine vs. CS-Mine

3.6. Remarks

From the analysis of the sequential pattern mining algorithms, it has been found that the WAP-tree based mining algorithms outperform the traditional sequential pattern mining algorithms (e.g., Apriori-based algorithms) due to the elimination of expensive candidate generation-and-test and the reduction of the times of database scan. The two WAP-based algorithms, PLWAP-Mine and CS-Mine, is the outstanding sequence pattern mining algorithms. These two algorithms are further compared by applying them in the Web-page recommender system using the three real world Web access sequence datasets. The comparison results show that PLWAP-Mine or CS-Mine can be applied effectively in the Web-page recommender system, and CS-Mine is efficient in cases of regularly long sequences and small support threshold, while PLWAP-Mine is always reliable to generate complete frequent patterns but its execution time is long. This means CS-Mine is not appropriate to discover highly frequent item patterns.

Generally speaking, the following points may be considered when mining sequential patterns from Web logs in Web-page recommender systems. If the length of WAS and the speed are not important, it is better to use PLWAP-Mine algorithm for guaranteed quality. If the sequence length of dataset does not considerably fluctuate, e.g., the majority of sequences are long, as in the case of the Cezeife dataset, CS-Mine would be preferred. Otherwise, the PLWAP-Mine would be the preferred one to use. Although CS-Mine can result in faster and more reliable sequential patterns when the support threshold becomes small, the speed of algorithms is not important in the off-line phase in Web-page recommender systems. Hence, PLWAP-Mine is preferred to be used in the off-line phase to guarantee the completeness of frequent pattern sets extracted for Web-page recommendation in the SWRS of this research.

3.7. Summary

This chapter has focused on the WUM process and how to make Web-page recommendation using a WUM technique as the part of the SWRS of this research. This chapter has contributed to making a decision on selecting the best efficient mining technique which is used to discover FWAP, the core Web usage knowledge in the SWRS. PLWAP-Mine is selected as it can satisfy the issues of execution time, memory space, and recommendation accuracy, compared with the other sequence mining techniques. The next chapter will focus on domain knowledge discovery and representation as other part of the SWRS.

Chapter 4.

DOMAIN ONTOLOGY MODELLING FOR WEB-PAGE RECOMMENDATION

4.1. Introduction

As introduced in Chapter 2, ontology is a knowledge representation technology whose implementation can be machine-understandable using the ontology language, such as OWL. Ontology defines the concepts and their associations in an application domain. In the context of Web-page recommendation, it is necessary to have an ontology that expresses the meaning of Web-pages for better understanding Web usage patterns and discovering frequently viewed domain terms for supporting more effective Web-page recommendations.

The Web usage knowledge can be discovered from Web usage data through unsupervised learning processes, such as sequential pattern mining techniques, but without the semantics of Web-pages, the discovered knowledge are limited in supporting Web-page recommendation, such as no alleviation to the "new page" problem. A domain ontology is really useful to enhance a Web-page recommendation process by adding semantics to Web-pages, but how to build an effective domain ontology for Web-page recommendations is always a big challenge. The study presented in this chapter builds a domain ontology of Web-pages of a website that can be used to interpret the semantics of Web-pages. This chapter proposes a domain ontology model that represents the domain concepts, Web-pages, and the relations among them for a given website to support semantic-enhanced Web-page recommendation and also presents a novel method to build such a domain ontology for a website.

For the rest of the chapter, Section 4.2 presents the domain ontology model of a website. Section 4.3 proposes a new method for domain ontology modelling. Section 4.4 presents

reasoning algorithms for the domain ontology model of Web-pages. Section 4.5 shows a case study of developing a domain ontology of Microsoft's website. Section 4.6 evaluates and discusses the resulting ontology model. Section 4.7 summarizes this chapter.

4.2. Domain Ontology Model of a Website

In the context of Web-page recommendation, we build the domain ontology of Web-pages of a given website based on the visited Web-pages to represent the domain concepts (general domain terms), the relationships between the concepts with constraints, the instances of concepts (specific domain terms), Web-pages, and the links between Web-pages and specific domain terms.

4.2.1. Assumption

Our assumption is that the titles of Web-pages contain important information about the content of the pages, in other words, a Web-page title contains the keywords that embrace the semantics of the Web-page. For example, the following fragments from the Web-page of a subject offered by the University of Technology, Sydney (UTS) (Figure 4-1) show the metadata of this page, such as the page title and its author:

```
<title> UTS: 31241 3D Computer Animation - Information Technology,
UTS Handbook </title>
<meta name="author" content="University of Technology, Sydney" />.
```

This Web-page presents the information about a subject that teaches 3D computer animation offered by UTS. Its title "`31241 3D Computer Animation - Information Technology, UTS Handbook`" describes that this Web-page is about a university study program on 3D computer animation with a unique identification code "31241", which is run by UTS. It can be seen that the title exactly represents what this Web-page is about.

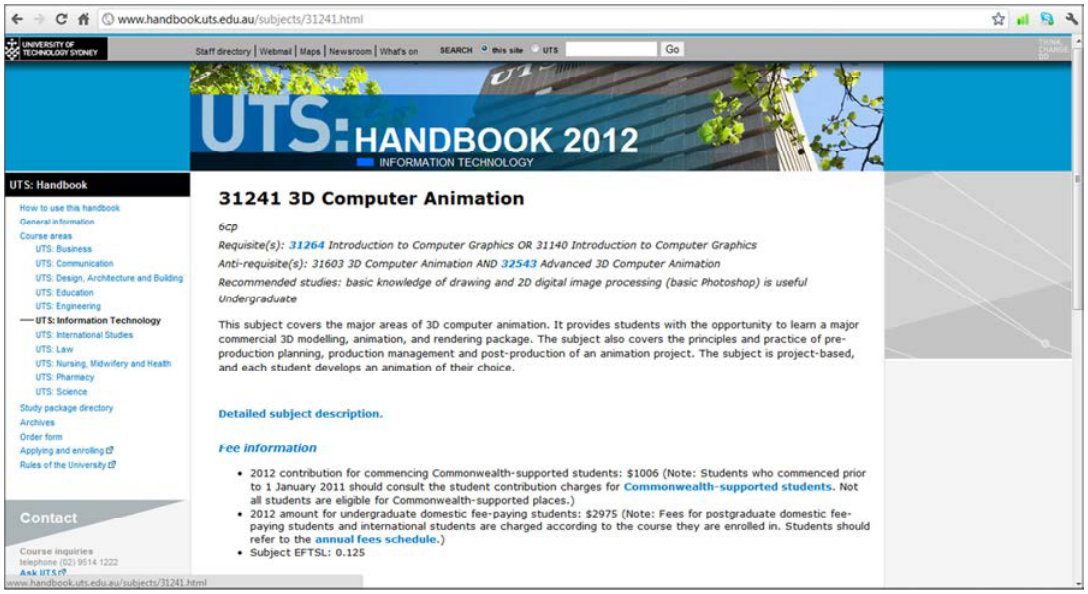


Figure 4-1: Sample Web document

The rationale behind this assumption can be seen from two aspects. One aspect is from the Web-page features. Generally speaking, a Web-page contains a collection of objects (represented by HTML tags) documented in metadata, which is data about data. Metadata embraces the core elements of title, meaning, descriptive context, structure, and overall context of a Web-page. By analysing the metadata, such as Web-page title, the meaning of a Web-page can be understood and captured. Metadata of a Web-page is usually included in the document head of the Web-page and can be retrieved by a Web crawler. The other aspect is from the professional practice in Web development. Professional Web developers need to follow a convention using a set of standard metadata tags to design Web documents. In well-designed Web-pages, the TITLE tag should contain the meaningful keywords which are relatively short and attractive to support Web search or crawling. In practice, the terms in page titles are usually given higher weights by search engines, such as Google (Liu 2011a; Markov & Larose 2007a). Consequently, website developers often define the Web-page titles very seriously because they want their Web-pages to be correctly identified during Web search or crawling and use the Web-page titles to convey accurate information about the Web-page.

Therefore, it is reasonable to make use of the metadata of accessed Web-pages discovered from the Web usage data, namely Web-page titles, to discover domain knowledge for domain ontology modelling of Web-pages within a given website and to support effective Web-page recommendation.

4.2.2. Domain Ontology Model of a Website

According to the definitions of an ontology in Section 2.3.1, this study defines a domain ontology model of a website as follows:

Definition 4.1 **(Domain ontology model of Web-pages - DomainOntoWP)** A domain ontology structure of a website is defined as a four-tuples: $O_{man} := < \mathcal{C}, \mathcal{D}, \mathcal{P}_{MAN}, \mathcal{A} >$, where $\mathcal{C}$ represents terms extracted from the Web-page titles within the given website, $\mathcal{D}$ represents the Web-pages of the website, $\mathcal{P}_{MAN}$ represents properties defined in the ontology, and $\mathcal{A}$ represents axioms, such as, an instantiation axiom assigning an instance to a class, an assertion axiom assigning two instances by means of a property, a domain axiom for a property and a class, and a range axiom for a property and a class. In details, $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{P}_{MAN}$ are further divided into sets:

$\mathcal{C} = C \cup T_{man}$ comprises a set of general domain terms (concepts) C, and a set of specific domain terms (instances of the concepts) T_{man},

$\mathcal{D} = SemPage \cup D$ comprises class *SemPage* which represents Web-page instances, and a set of Web-pages D,

$\mathcal{P}_{MAN} = R_{man} \cup A_{man}$ comprises a set R_{man} of the relations between terms (R_c) and the relations between terms and Web-pages (R_p), and a set of attributes A_{man} defined in the ontology. In particular, R_c will be specified depending on the application domain. $R_p = hasPage \sqcup isAbout$, where the 'hasPage' relation states that a domain term may have some Web-pages, and the 'isAbout' relation is the inverse of the 'hasPage' relation. That means each domain concept class has the 'hasPage' object property referring to class *SemPage*, and class *SemPage* has the 'isAbout' object property referring to the domain concept classes.

This ontology structure is constructed at three levels: 1) *General level*, which holds the concepts that present the general domain terms of Web-pages and relationship definition sets, e.g. *C*, *SemPage,* and P_{MAN}; 2) *Specific level*, which holds the specific domain terms that are usually related to specific applications within the domain, e.g. T_{man}, and the relationships between them; 3) *Web-page level*, which holds all the Web-pages within the given website, e.g. *D*, and the association relationships between Web-pages and terms. The general level is presented as an ontology schema, and the specific and Web-page levels are presented as ontology instances. Such an ontology model supports modular development, scalability and reusability at different levels.

The general terms within a domain are usually stable and have little changes over the time while the specific terms can increase frequently with the evolution of the Web-page. For instances, when a new Web-page is generated in the site, its title can very likely contain specific terms that are part of existing general terms, but not in the domain ontology. With this ontology structure, these specific terms can be easily added into the ontology at the specific level and the Web-page can be included easily at the Web-page level.

In applying this ontology model to real world websites, the elements at the general and specific levels can be developed using conventional ontology modelling techniques. Within the context of Web-page recommendation, this study takes advantages of metadata of Web-pages, especially the Web-page titles, to determine the domain terms, including the general and specific terms, to save development effort. Updating elements at the Web-page level can be very challenging because there are often a numerous number of Web-pages in a website, and establishing the links between specific terms and Web-pages can be daunting. In order to circumvent this potential difficulty, a smart algorithm is developed to automatically add new Web-page instances into the domain ontology and map the Web-pages to the domain terms using keyword expressions. Details about how to build such an ontology model of a given website will be detailed in the following sections.

4.3. New Method for Domain Ontology Modelling of a Website

Construction of domain ontologies is a challenging and time-consuming endeavour. Therefore, this section proposes a new method of modelling domain ontology that would reduce human effort in ontology construction. According to (Grimm et al. 2011), a generic methodology of ontology engineering comprises three main steps: requirement analysis, conceptualization, and implementation. The new method developed for modelling domain ontology of a website consists of three parts that correspond to the main steps in the general methodology. The flow diagram of this new method is shown in Figure 4-2.

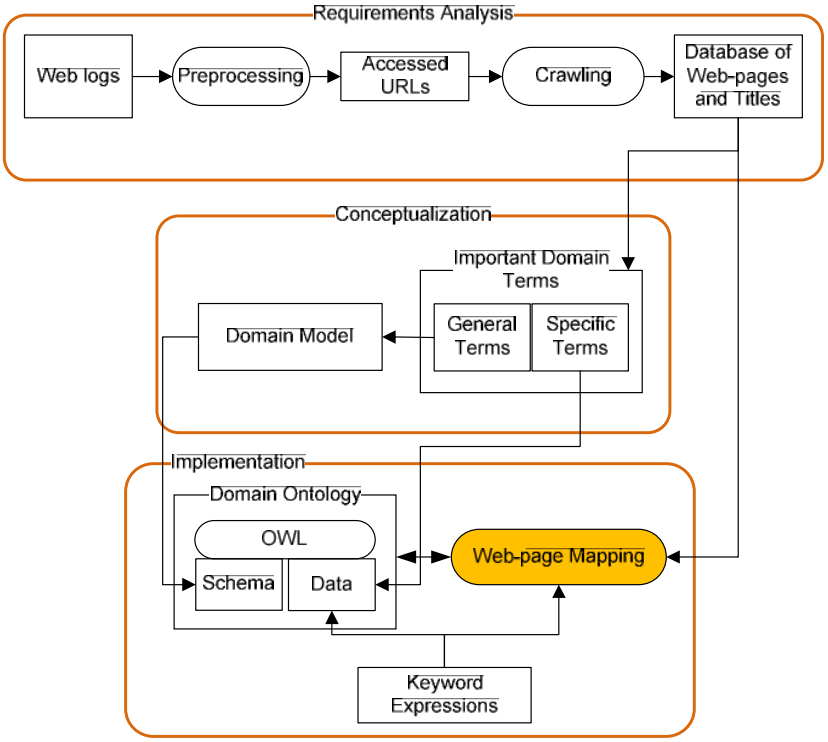


Figure 4-2: The flow diagram for domain ontology modelling

(1) Requirements Analysis

In the context of Web-page recommendations, the goal of domain ontology of a website is to represent the semantics of the Web-pages within the given website. Its scope is to cover

the content of those Web-pages. In this study, we use Web-page titles to represent the content of Web-pages. See Section 4.2.1 for the rationale.

In order to model the domain knowledge related to user interests for supporting Web-page recommendation, and to reduce the processing load, this study focuses on the Web-pages that have been visited by users (referred to as accessed or visited or viewed Web-pages), which can be obtained from Web server logs.

As described in Chapter 3, Web logs record user sessions on a daily basis, which include information about users' Web-page navigation activities. Web logs need to be cleaned to remove unnecessary information, such as advertisement, through a pre-processing process, as presented in Section 3.2. The visited Web-pages in each user sessions can be extracted from the cleansed Web logs. The visited Web-paged are identified by their Web addressed or URI (universal resource identifier) or URL (universal resource link).

After the URLs of visited Web-pages are extracted from the Web logs, the titles of these Web-pages can be retrieved using Web crawling technique. Web crawling (Markov & Larose 2007b) is used to retrieve Web content based on URLs. The goal of Web crawling is to explore a collection of linked Web documents by fetching Web-pages at many levels based on one URL. The first step is address resolution which converts the symbolic Web address of a page into an IP address to locate the server. After a Web-page is fetched, the hyperlinks (URLs) will be extracted. The new URLs will be traced by the crawler respectively. In this study, since the URLs of visited Web-pages are known, a simple Web crawler is developed to retrieve the titles of these Web-pages. The crawler is designed to look for the TITLE tags in Web documents and to retrieve the corresponding values as the titles. The output of the crawler is the set of titles of the all viewed Web-pages of the given website.

(2) Conceptualization

Based on the titles of visited Web-pages, domain experts can identify the domain terms for the website. From general to specific or vice versa, the domain experts need to classify the terms to be general or specific and the *belongs* relations between the general and specific

terms. The general terms will become the domain concepts (C), the specific terms become the instances of the general terms (T_{man}) and the *belongs* relations become the instantiation axioms between the instances and the domain concepts in the domain ontology.

Domain experts also need to build the ontology schema at the general level by identifying the relations between the domain concepts, and the associations between them and *SemPage*, i.e. R_{man}, and the attributes of the domain concepts and *SemPage*, i.e. A_{man}. A special attribute, namely keyword string, is required for each domain concept for mapping between domain terms and Web-pages in a later stage. How to define the keyword strings for each specific term will be explained later on.

With C, *SemPage,* R_{man} and A_{man}, the design of ontology schema is accomplished. This schema can usually be expressed as a diagram which is similar to the entity-relationship diagram for a relational database. An example is given in the case study in Section 4.5.

(3) Implementation

The domain ontology including the schema and data can be easily implemented using a popular ontology language, i.e. OWL, which is designed to represent rich and complex knowledge about things, groups of things, and relations between things. Protégé[8] as a commonly used OWL-based ontology editor can be used to implement such an ontology at the general and specific levels.

Special effort has been devoted to define the keyword strings for each specific term. Table 4-1 shows the BNT-like grammar, which are used as the composition rules of keyword expressions.

[8] http://protege.stanford.edu/; Accessed November, 2012

Table 4-1: Keyword expressions

Keyword Expressions
K := <kw> | [<kw>] I := K | %syn%K A := I | !I Exp := A(&&A)* where, - <kw>: a keyword - [<kw>]: a keyword in capitals - %syn%: the synonyms of keyword K - !: not contain keywords specified in I - &&: AND operator - *: REPEAT operator for the expression in the two brackets () - Exp: a keyword expression

There are four rules in this grammar set. The first rule is for specifying a keyword or a keyword in capitals, e.g. <"internet"; "[IE]"> is two keywords for specific term "internet"; The second rule is for specifying a list of keywords specified by the first rule or the synonyms of a keyword, e.g. <"%syn%Development"> is a keyword string for specific term "development"; The third rule is for specifying the list of keyword strings that can be specified by the second rule or a keyword string that doesn't contain the keyword string specified by the second rule; The last rule is for specifying the keyword strings that are formed by an expression involving one or more keyword strings that are specified by the third rule. It is noticed that OR operator is not included in the syntax because keyword strings can be input separately as a set of attribute values in the ontology editor tool, i.e. Protégé, and that set is handled as the union of keyword strings.

Based on this set of grammar rules, the domain experts or professional developers can easily define keyword strings for specific terms in the ontology. These keyword strings will be used to map the corresponding Web-pages with the specific terms later on.

Once the ontology schema has been implemented and specific terms have been added into the ontology in the ontology editor, an ontology file can be easily generated by the editor. This ontology file will be used to populate with Web-pages and the associations between Web-pages and specific terms at the Web-page level.

The details about how the Web-page mapping is carried out is shown in Figure 4-5 as part of the flow diagram of domain ontology modelling. In which, the input data includes

the domain ontology file exported from the ontology editor in OWL, and the Web-page titles. The keyword expressions are used for parsing the keyword strings of each domain terms in the domain ontology. WordNet is used to query for the synonyms of keywords during the mapping process. The used WordNet is the core WordNet 3.0 database in MySQL[9], which contains total of 147278 unique noun, verb, adjective, and adverb strings and is convenient to operate in Java programs.

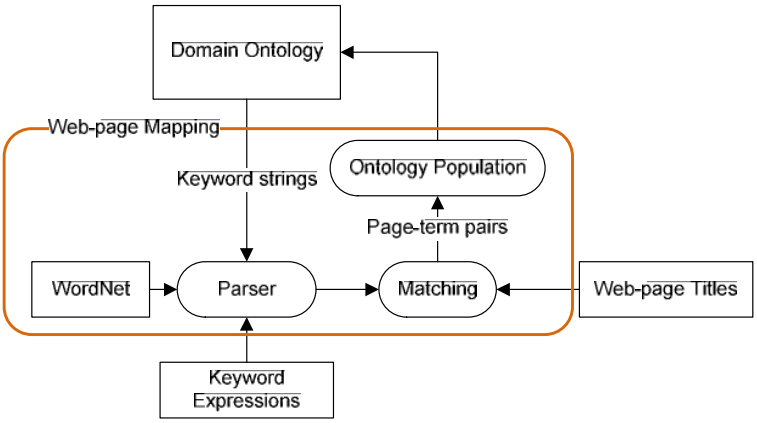


Figure 4-3: Web-page mapping

WordNet is an external digital dictionary of common English words that can be used to identify concepts in different application domains. WordNet has been developed under a research project at the Princeton University with the purpose of producing the combination of dictionary and thesaurus, and supporting automatic text analysis. It was designed to build connections between four types of Parts of Speech (POS): noun, verb, adjective and adverb. The basic concept in WordNet is SYNSET (SYNonym SET), which represents a specific meaning of a word. A synset contains a set of words with synonymous meanings. A synset contains one or more word senses and each word sense belongs to exactly one synset. In turn, each word sense has exactly one word that represents it lexically, and one word can be related to one or more word senses. The concept of each synset is defined shortly in text. For example, the words “night”, “night-time” and “dark” constitute a single

[9] http://wnsqlbuilder.sourceforge.net/download.html; Accessed January, 2011

synset that has the following gloss: "the time after sunset and before sunrise while it is dark outside". Synsets are connected through relations, such as opposites, IS-A, PART-OF, etc.

WordNet has been converted to RDF/OWL representation[10], so that it can be adopted in Semantic Web applications for the annotation and retrieval of information in knowledge management systems. More than 100,000 words are defined by WordNet, so WordNet is a useful dictionary for building an ontology of some application domains.

The mapping program is implemented in Java programming language within the NetBean developing platform. The core part in this program is a specially designed smart algorithm which finds the mapping between the specific terms and Web-pages based on the keyword strings of specific terms and the Web-page titles with the support of WordNet. The algorithm, named as Algorithm 4-1, is shown in below.

Algorithm 4-1: Mapping Web-pages to domain terms using the keyword expressions

```
For each Web-page{
    For each concept in the domain ontology{
        For each instance of the concept{
            Check If the title of the Web-page meets the keywords of the domain concept instance:
                Using the rules of keyword expressions to parse the keyword strings
                Using WordNet to query synonyms if necessary
            Then create a new SemPage instance of the Web-page, and map it to the domain concept
                instance via the 'isAbout' and 'hasPage' object properties
        }
    }
}
```

Based on the composition rules of keyword expressions, the system firstly parse the keyword strings of each specific domain term in the domain ontology, where WordNet is used to identify synonyms if applicable, to return the specific keywords for each term. Secondly, the specific keywords for each term are matched up with a Web-page title, and return the matching page-term pairs where one Web-page whose title meets the specified keyword strings of one term. Thirdly, if a match is found between a specific term and a Web-page title, a new Web-page instance will be automatically created with the following property values: title is the Web-page title and the 'isAbout' property is referred to the matched term. And then a new value of 'hasPage' property of the matched term will be set

[10] http://www.w3.org/TR/2006/WD-wordnet-rdf-20060619/; Accessed April, 2011

to point to the matched Web-page. If no match is found for a Web-page title, new specific terms will be generated based on the keywords in the Web-page title to be added into the concept has most relevant terms in the domain ontology, and then the Web-page instance is created and mapped to these terms.

After the mapping process is completed, the ontology model of this given website is ready to be used in Web-page recommendation systems. It supports automated reasoning processes due to its feature of being machine-understandable.

4.4. Reasoning Algorithms for the Domain Ontology Model of Web-pages

Within the context of Web-page recommendation, the domain ontology of Web-pages, namely DomainOntoWP, can be used to interpret the Web-pages or find Web-pages for a given specific domain term in an automated fashion. Based on Definition 4.1, this section develops two reasoning algorithms to perform these two tasks. These two algorithms are also used for Web-page recommender system in later chapters.

The first reasoning algorithm, named as Algorithm 4-2, is proposed to query for the domain terms (topic) of a given Web-page $d \in D$ by retrieving domain concept instances which are associated with the *SemPage* instance d via the *isAbout* object property. This algorithm is referred to as function $\boldsymbol{Topic_{man}(d)}$ in the later recommendation algorithms.

This algorithm can be formally expressed in Description Logics as $q_1(x)$:- $C(x)$, $hasPage(x, d)$.

Algorithm 4-2: Query about domain terms of a given Web-page

Input:
  d (PageID)
  O (DomainOntoWP)
Output: T (list of domain terms)
Process:
  Traverse through O to get the *SemPage* instance whose PageID is d
  Set T = domain concept instances associated with Web-page d via the 'isAbout' object property
  Return T

Note that if the given page, in the first reasoning algorithm, is not in the DomainOntoWP, it will be processed to be mapped to domain terms in the DomainOntoWP using Algorithm 4-1. Subsequently, the terms of this page are returned.

The second reasoning algorithm, as Algorithm 4-3, is proposed to query for the Web-pages of a given domain term $t \in T_{man}$ by retrieving *SemPage* instances which are mapped to the domain concept instance t via the 'hasPage' object property. This algorithm is referred to as function $\boldsymbol{Page_{man}(t)}$ in the later recommendation algorithms. This algorithm can be formally expressed in Description Logics as $q_2(x)$:- $SemPage(x)$, $isAbout(x, t)$.

Algorithm 4-3: Query about pages mapped to a given domain term

Input:
t (domain term)
O (DomainOntoWP)
Output: DS (set of pages mapped to *t*)
Process:
Traverse through *O* to get the domain concept instance whose name is *t*
Set *DS* = the set of *SemPage* instances mapped to *t* via the 'hasPage' object property
Return *DS*

4.5. Case Study: Development of a Domain Ontology of the Microsoft Website

The Microsoft anonymous Web data which is the Web usage data of the MS website created in 1998 is used as a case study of domain ontology construction in this study. The dataset is able to be found from http://kdd.ics.uci.edu/databases/msweb/msweb.html. Based on the method of domain ontology modelling presented in Section 4.3, the domain ontology of the MS website is developed in the following steps.

4.5.1. Requirements Analysis

Using the MS Web dataset, the Web-page titles and paths of the MS website are available as illustrated in Table 4-2.

Table 4-2: A Sample MS Web dataset

Title	Path
MS Access Support	/msaccesssupport
SQL Support	/sqlsupport
SQL Server	/sql
MS Office	/msoffice
MS Office News	/msofc
MS PowerPoint	/mspowerpoint
MS Project Support	/msprojectsupport
MS Excel Support	/msexcelsupport

This study focuses on the application scenario of MS software products, such as MS Office, Windows Operating System, and Database, so domain terms of this interest are identified in the next sub-section.

4.5.2. Conceptualization

With the given dataset, meaningful terms are extracted by removing stop words, e.g., "the", "a", and "for", or invalid words from the Web-page titles. For example, terms which are extracted from the sample dataset in Table 4-2 are "MS", "Access", "Support", "SQL Server", "Office", "News", "PowerPoint", "Project", and "Excel". It is possible for some extracted terms to share same features, so they are better to be the instances of a concept rather than standalone concepts. As the scope of the domain has been stated in the requirements analysis, the considered domain concepts of the MS website are *Manufacturer*, *Application*, *Product*, *Category*, *Solution*, *Support*, *News*, *Misc*, and *SemPage*. In which, the concept *SemPage* refers to Web-pages, and other concepts refer to terms used in the MS website. Table 4-3 shows the general domain concepts along with their (specific) domain terms.

Table 4-3: Domain concepts and corresponding domain terms

Domain Concept	Domain Terms
Manufacturer	Microsoft
Application	Database, Office, Visual Studio, etc.
Product	Excel, FontPage, MS Access, MS Office, etc.
Category	Software, Hardware, etc.
Solution	MS Solution, etc.
Support	Developer, Education, Network, Server, etc.
News	MS News, etc.
Misc	Games, Sport, etc.

In order to specify the domain model of the website, taxonomic and non-taxonomic relationships are necessary to be defined. For taxonomic relationships, we start with the concept *Application* and *Product*. Considering the 'consistsOf' relation which indicates that a concept comprises a number of parts which are also concepts, we particularly have an *application* may consist of some *sub-applications*. For instance, an *application* software of *Office* has some *sub-applications* including *Word*, *Access*, *Power Point*, etc. Considering the 'includes' relation, we have a *product* may include some *sub-products*, e.g., *Office* software *product* includes *sub-products*, such as *MS Word*, *MS Access*, and *MS Power Point*. Considering the 'belongsTo' relation, we have a *product* may *belong to* one or some certain *category*, e.g. *software*, *hardware*, *entertainment*, or *service*.

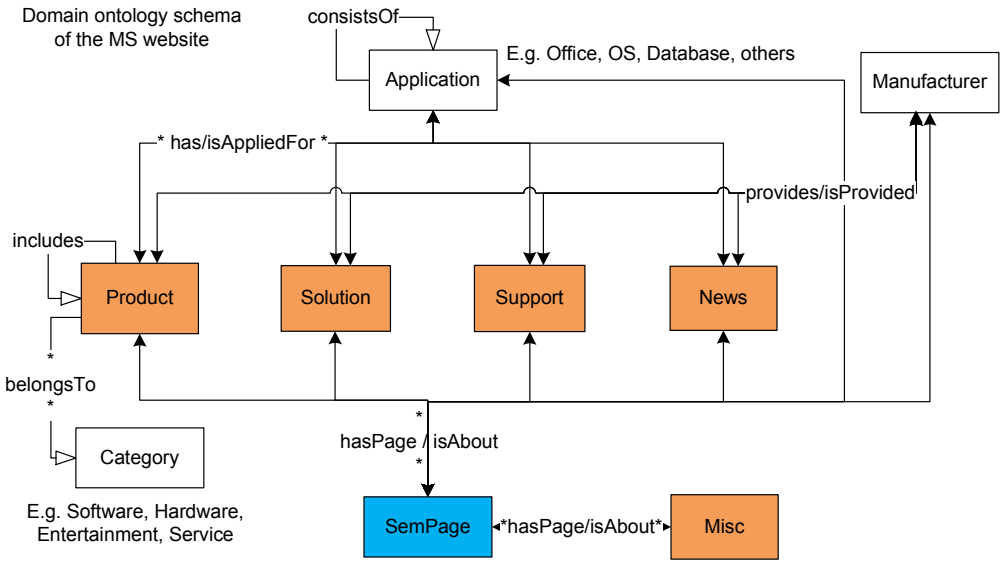


Figure 4-4: Domain ontology schema of the MS website

Regarding the non-taxonomic relationships, the relationship types, e.g. self-referencing, 1-M, and M-N relationships, which are often used in a relational database except for the relationships between a super set and a sub set are considered. In the MS website example, the main types of non-taxonomic relationships are listed as below.

- The 'provides' relation describes the M:N relationship between concept *Manufacturer* and concepts *Product*, *Solution*, *Support*, and *News.* For example, the

MS *manufacturer* might provide some *products*, e.g. MS Office, or some *solutions*, e.g. MS Solutions. The 'isProvided' relation is the inverse of the 'provides' relation.

- The 'has' relation describes the M:N relationship between concept *Application* and concepts *Product*, *Solution*, *Support*, and *News*. For example, the Office *application* might have some *products*, e.g. MS Office, MS Project, etc., or some *supports*, e.g. MS Office Support, MS Project Support, etc. The 'isAppliedFor' relation is the inverse of the 'has' relation.
- The 'hasPage' relation describes the M:N relationship between a domain concept, such as *Application* or *Product*, and the concept *SemPage*. For example, the MS Word *application* has some Web-pages describing its general information and features. The 'isAbout' relation is the inverse of the 'hasPage' relation, which means when we define a page *about* a certain term instance, that term instance *has* the page as its object property value.

Combining the taxonomic and non-taxonomic relationships between the concepts, we have a domain model as the ontology schema of the MS website as shown in Figure 4-4. To formalize this domain knowledge representation model in Description Logics, the domain ontology model of Web-pages for the MS website is defined in the following.

Domain ontology model of Web-pages for the MS website

Based on the domain ontology model of Web-pages in Section 4.2.2, the domain ontology model for the MS website can be specified. In particular, three main constitutes of the ontology: (1) T_{man} is a set of domain terms in the given website and classified into the domain concepts, e.g. Manufacturer, Application, etc., (2) D is a set of Web-pages in the given website, and (3) R_{man} is a set of association relations including the taxonomic and non-taxonomic relationships in the domain model of the given website, are detailed as follows:

$T_{man} \doteq Manufacturer \sqcup Application \sqcup Product \sqcup Category \sqcup Solution \sqcup Support \sqcup News \sqcup Misc,$

$D \doteq SemPage$, and

$R_{man} \doteq consistsOf \sqcup includes \sqcup belongsTo \sqcup provides \sqcup isProvided \sqcup has \sqcup isAppliedFor \sqcup hasPage \sqcup isAbout.$

The knowledge base of the model is described as follows:

$\exists consistsOf.Application \sqsubseteq Application,$

$\exists includes.Product \sqsubseteq Product,$

$Product \sqsubseteq \exists belongsTo.Category,$

$Manuafacturer \sqsubseteq \exists provides.(Product \sqcup Solution \sqcup Support \sqcup News),$

$(Product \sqcap Solution \sqcap Support \sqcap News) \sqsubseteq \bot,$

$isProvided \equiv provides^{-},$

$Application \sqsubseteq \exists has.(Product \sqcup Solution \sqcup Support \sqcup News),$

$isAppliedFor \equiv has^{-},$

$SemPage \sqsubseteq \exists isAbout.(Manuafacturer \sqcup Application \sqcup Product \sqcup Solution \sqcup Support \sqcup News \sqcup Misc)$, and

$hasPage \equiv isAbout^{-}.$

4.5.3. Implementation

OWL is used to build the domain ontology of Web-pages for the MS website, namely **DomainOntoWP**, based on the specified domain model. Protégé is used to edit the ontology schema. Figure 4-5 lists partial OWL implementation of the object properties which represent the links between concept classes in the ontology. The *inverseOf* property is used to enforce the constraint that a relation is the inverse of another relation. The *unionOf* construct is used to express a set operation, namely the union of a collection of concept classes.

```
<owl:ObjectProperty rdf:ID="consistsOf">
  <rdfs:range rdf:resource="#Application"/>
  <rdfs:domain rdf:resource="#Application"/>
 </owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="includes">
  <rdfs:range rdf:resource="#Product"/>
  <rdfs:domain rdf:resource="#Product"/>
 </owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="belongsTo">
  <rdfs:range rdf:resource="#Category"/>
  <rdfs:domain rdf:resource="#Product"/>
 </owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="provides">
     <rdfs:domain rdf:resource="#Manufacturer"/>
     <owl:inverseOf rdf:resource="#isProvided"/>
     <rdfs:range>
        <owl:Class>
           <owl:unionOf rdf:parseType="Collection">
              <owl:Class rdf:about="#News"/>
              <owl:Class rdf:about="#Product"/>
              <owl:Class rdf:about="#Solution"/>
              <owl:Class rdf:about="#Support"/>
           </owl:unionOf>
        </owl:Class>
     </rdfs:range>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="has">
     <rdfs:domain rdf:resource="#Application"/>
     <owl:inverseOf rdf:resource="#isAppliedFor"/>
     <rdfs:range>
        <owl:Class>
           <owl:unionOf rdf:parseType="Collection">
              <owl:Class rdf:about="#News"/>
              <owl:Class rdf:about="#Product"/>
              <owl:Class rdf:about="#Solution"/>
              <owl:Class rdf:about="#Support"/>
           </owl:unionOf>
        </owl:Class>
     </rdfs:range>
  </owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasPage">
  <owl:inverseOf rdf:resource="#isAbout"/>
  <rdfs:domain>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#Product"/>
     <owl:Class rdf:about="#News"/>
     <owl:Class rdf:about="#Misc"/>
     <owl:Class rdf:about="#Application"/>
     <owl:Class rdf:about="#Support"/>
     <owl:Class rdf:about="#Manufacturer"/>
     <owl:Class rdf:about="#Solution"/>
    </owl:unionOf>
   </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="#SemPage"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="isAbout">
  <owl:inverseOf rdf:resource="#hasPage"/>
  <rdfs:domain rdf:resource="#SemPage"/>
  <rdfs:range>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#News"/>
     <owl:Class rdf:about="#Application"/>
     <owl:Class rdf:about="#Misc"/>
     <owl:Class rdf:about="#Support"/>
     <owl:Class rdf:about="#Manufacturer"/>
     <owl:Class rdf:about="#Product"/>
     <owl:Class rdf:about="#Solution"/>
    </owl:unionOf>
   </owl:Class>
  </rdfs:range>
</owl:ObjectProperty>
```

Figure 4-5: Specification of object properties in the domain ontology of the MS website

After building the ontology schema, the specific domain terms as instances of the concepts in the specific domain of the MS website, such as Microsoft, Word, Excel, and relevant terms, are manually added into the corresponding concepts in the ontology. For example, concept instances Word, Excel, FrontPage, etc. are added into concept class *Product* (Figure 4-6).

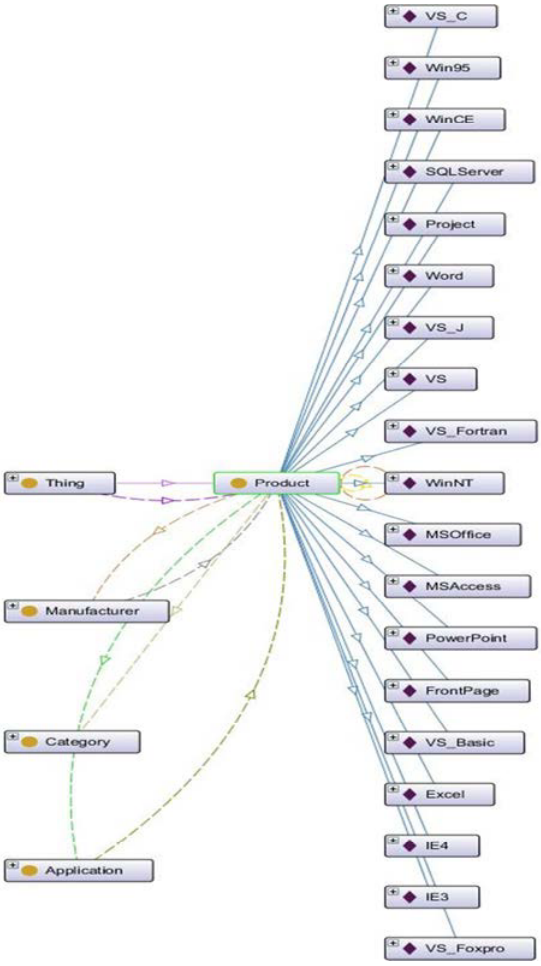


Figure 4-6: The part of the domain ontology: *Product* instances

For each term instances which may have some Web-pages, a *Keyword* property is added into each domain concept class in the ontology. *Keyword* property values are strings described in the syntax of keyword expressions. *Keyword* strings should be specific terms being about the concept instances. If a set of keyword strings are added, they will be processed as the union of keyword strings like the OR operator. To obtain the maximum profit of keyword-based Web-page mapping, we need to specify keyword strings properly for each concept instance to ensure that relevant Web-pages can be mapped to it. This is decided by the designer.

The Web-page mapping algorithm is used to automatically generate Web-page instances and map them to domain terms. Based on the composition rules of keyword expressions (Table 4-1), this algorithm can parse the keyword strings of each domain term, and determine Web-pages whose titles meet the specified keyword strings to be mapped to the term.

For example, the *Support* concept has instances, such as 'Development', 'Network' and 'OtherSupport'. The specified keywords of instance 'Development' are "%syn%Development", meaning the synonyms of 'Development'; so Web-pages whose titles contain terms, e.g. 'Development' or 'Developing', should be mapped to this instance. Table 4-4 shows the IDs and titles of some Web-pages mapped to instances 'Development', 'Network' and 'OtherSupport'. To avoid overlapping between the Web-pages of two or more domain terms, we can strictly specify keyword strings as the cases of 'Network' and 'OtherSupport'. Keyword string "support && !network" means Web-pages whose titles contain "support" but not "network" belong to concept instance 'OtherSupport'. Thus, the title of Web-page 1049 "**Support Network** Program Information" contains both keywords "support" and "network", so this page is mapped to concept instance 'Network' only.

Table 4-4: Mapping some Web-pages to some domain terms based on the specified keyword strings

Concept	Specific Term	Keyword strings	Mapped Web-pages
Support	'Development'	"%syn%Development"	1012: Outlook **Development** 1027: Internet **Development** 1187: ODBC **Development** 1236: **Developing** for Global Markets …
	'Network'	"network"; "security"; etc.	1049: **Support Network** Program Information 1113: Internet **Security** Framework …
	'OtherSupport'	"support && !network"	1085: Exchange **Support** 1132: MS Money **Support** 1162: IIS **Support** …
Product	'Word'	"MS Word"	1060: **MS Word** 1052: **MS Word** News 1135: **MS Word** Support 1253: **MS Word** Development …

On the other hand, a Web-page instance might also be mapped to two or more domain concept instances when its title contains the keywords of those concept instances. For example, considering Web-page 1253 with title "MS Word Development", this page might be mapped to the *Product* instance 'Word' and the *Support* instance 'Development' (Table 4-4).

Furthermore, if the title of a Web-page does not meet any keyword strings of all domain terms of a domain concept, such as, *Support* and *News*, a new term instance is created based on keywords in the title and the existing domain terms. For instance, we have the *Application* concept including instances 'Office' and 'appWord', and want to classify page 1253 into a *Support* instance which is applied for an *Application* instance 'appWord'. Hence, we can define the keyword strings of the *Support* instance 'Development' as "%syn%Development && !Word", so that the title of page 1253 does not meet that keyword string. A new *Support* instance "appWord_Development" should be created for 'appWord', and associated with page 1253.

Figure 4-7 depicts part of the domain ontology presenting the *Product* instance 'Word' connected to the *Application* instance 'appWord', Web-page 1060 connected to 'Word', Web-page 1253 connected to instance 'Word' and the *Support* instance 'appWord_Development', Web-page 1135 connected to instance 'Word' and a *Support* instance 'appWord_OtherSupport', and Web-page 1052 connected to instance 'Word' and a *News* instance 'appWord_MSNews'.

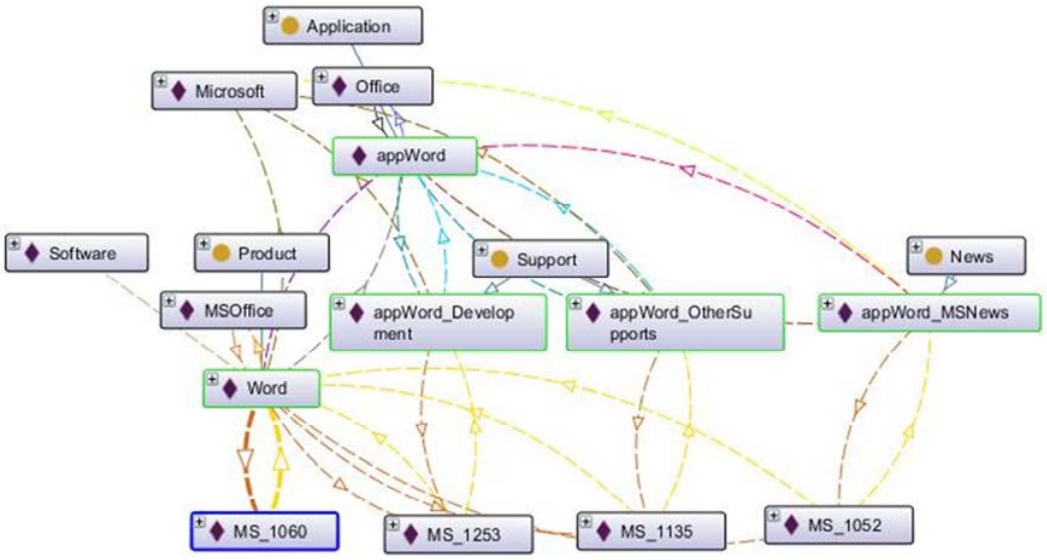


Figure 4-7: The part of the domain ontology: *Product* instance - Word

Based on this domain ontology built in OWL, we can perform the reasoning algorithms for supporting Web-page recommendation. For the example in Figure 4-6, we can query for the domain terms of pages 1060 and 1253, as follow:

$Topic_{man}$(‘MS_1060’) = {‘Word’},
$Topic_{man}$(‘MS_1253’) = {‘Word’, ‘appWord_Development’}.

That means pages 1060 and 1253 are about “Word”, and “Word Development”, respectively. We can also query for the pages of domain term “Word”, as follows:

$Page_{man}$(‘Word’) = {‘MS_1060’, ‘MS_1253’, ‘MS_1135’, ‘MS_1052’}.

That means domain term “Word” has pages 1060, 1253, 1135, and 1052 mapped.

4.6. Evaluation and Discussion

This section explains the evaluation method of ontology model to verify the built domain ontology, and gives some discussions.

4.6.1. Evaluation Method

As presented in Section 2.3.9, the schema metrics and instance metrics, i.e., Relationship Richness (RR), Class Richness (CR), Class Instance Distribution (CID), Class Connectivity, Class Importance (Imp(C_i)), and Relationship Utilization (RU), are used to evaluate the domain ontology model.

Regarding ontology validation, the quality of the domain ontology can be assessed by evaluating how the ontology answers some queries of entities in the domain, such as Web-pages or term instances. In other words, the ontology can be validated by examining the reasoning algorithms across the domain ontology. Reasoning results can be evaluated by human, but it will be more convincing to apply this domain ontology to a practical system and to evaluate the system performance with/without the domain ontology. Therefore, the domain ontology validation will be elaborated in Chapter 7, making use of the domain ontology for the proposed SWRS.

The following sub-section presents an evaluation of the modelled domain ontology in terms of the defined metrics in Section 2.3.9.

4.6.2. Evaluation of Experimental Results, and Discussions

As described in the case study of domain ontology development for the MS website, there are nine defined concept classes and a number of created domain concept instances as well as added Web-page instances. Table 4-5 summarizes the ontology metrics of the constructed domain ontology based on the mentioned evaluation measures. For each class, the number of *instances*, *connectivities*, and *importances* are shown in the table. The values of RU, which equal 1, reflect that the all defined relationships are used.

Table 4-5: Evaluation of the domain ontology of the MS website

Class	# Instances	Connectivity	Importance	RU
Application	17	85	0.067460317	1
Category	4	0	0.015873016	1
Manufacturer	1	86	0.003968254	1
Misc	3	3	0.011904762	1
News	5	21	0.01984127	1
Product	19	148	0.075396825	1
SemPage	173	216	0.686507937	1
Solution	1	5	0.003968254	1
Support	29	151	0.115079365	1
SUM of Instances	252			
RR	1			
CR	1			
CID	55.24491			

The *class importance* shows that class *SemPage* is most important because more than 50% of instances in the ontology are Web-pages, and the second and third important classes are *Support* and *Product*. About *class connectivity*, class *SemPage* is the most connected class with 216 connections to other concept instances out of a total of 715 connections. The second and third most connected classes are *Product* and *Support* with 148 and 151 connections, respectively. This indicates that *Product* and *Support* are more focal classes than the remaining concept classes. It is reasonable because the built ontology focuses on software products and supports in the domain of the MS website.

As a whole, the value of RR is 1, that indicates that there is no the relationship of class-subclass because this relationship was not defined in the ontology schema. However, if we look at class *Application*, there is a recursive relation 'consistsOf' which means an application is able to consist of many sub-applications which might consist some sub-applications else. This definition allows us to easily extend concept *Application* in the future. Similarly, the 'includes' relation associates the *Product* concept instances, i.e. a product may include many sub-products. Sub-classes are not used in the case study because the 'consistOf' and 'includes' relations are more flexible to extend sub-concepts. Actually, these relations are like the inverse of the "Is-a" relationship type. In other words, these

relationships can form hierarchies, indicating that a super-concept can have one or more sub-concepts. Regarding the instance metrics, class richness and class instance distribution are measured. The CR value being 1 reflects that all designed classes have instances, which means no redundant class. About the CID of the domain ontology which indicates how instances are spread across the classes of the schema, the standard deviation in the number of instances per class is 55.24491.

To sum up, class *SemPage* is the most important class because it represents Web-pages which are able to describe various domain terms of the MS software products. This emphasizes the profit of automatically Web-pages mapping process when a huge amount of instances are Web-pages. Mapping Web-pages using the keyword expressions is an empirical method. Keyword strings are added into domain concept instances by the designer, so it is bias. However, its advantage is that the correctness can be optimized by changing the keyword strings of domain terms to have more Web-pages properly mapped. As a result, the built ontology has 97% of the mapped Web-pages accepted.

4.7. Summary

Although some domain ontologies have been built for domain knowledge representation in Web-page recommender systems, Web-pages might not be involved in most of existing domain ontologies. This chapter has, therefore, addressed how Web-pages should be presented with regard to a domain ontology, and how to develop a domain ontology for a website of software products. While most of existing ontologies are based on the "Is-a" relationship, it has been found that it is richer to use a variety of relationships for the ontology model. This chapter captured some relevant conceptual models (Dzemydiene & Tankeleviciene 2008; Gascuena, Fernandez-Caballero & Gonzalez 2006) to build an ontology which can be reused and shared with other applications. This chapter has proposed the ontology structure designed at three levels: general (domain concepts), specific (domain terms) and Web-page, allowing more flexibility in utilizing this ontological structure at different levels. The general level is less changed, so the domain model has been designed stably and enables changes at the specific and Web-page levels.

For example, we can update specific domain terms into domain concepts and define the keyword strings of each term for mapping Web-pages to the terms. The number of Web-pages is usually much more than the number of domain terms, so, the automatically mapping Web-pages to terms can significantly reduce manual effort, especially for large websites with many Web-pages. The important thing is this domain ontology can facilitate performing the algorithms reasoning about domain terms and Web-pages for semantically enhancing Web-page recommendation.

In summary, this chapter has shown that this domain ontology construction allows the ontology to be confident, rich and detailed because of basing on a domain expert's experience. The ontological description of domain terms and Web-pages allows the machine to understand the meaning of Web-pages by reasoning about terms associated with the Web-pages and vice versa, so this domain ontology is sufficient to support Web-page recommendation in recommender systems.

Chapter 5.

SEMANTIC NETWORK MODELLING FOR WEB-PAGE RECOMMENDATION

5.1. Introduction

Traditional ontology construction is a labour-intensive and time-consuming task and highly relies on human experts. Moreover, such constructed ontologies are often fixed to a specific domain of interest. This often leads to the difficulties of reusing existing ontologies. This makes it hardly possible to fully automate the knowledge acquisition process which includes knowledge discovery, knowledge base construction and knowledge utilization. With the rapid development of websites in quantity and quality, the manual construction of domain ontologies is no longer feasible for coping with the changeable websites. Therefore, it has become highly desirable to develop an efficient method to automate knowledge acquisition, representation and application.

As discussed in Section 2.3.4, depending on the purposes of knowledge representation, the knowledge can be conceptualized in various degrees of formality and can be described in the form of concept schemes, taxonomies, conceptual data models, or general logical theories (Grimm et al. 2011). Chapter 4 has proposed the new method of modelling a domain ontology as a conceptual data model and a new technique to build such an ontology model. The ontology structure has Web-page as a distinguishing concept and a pair of relations, namely 'isAbout' and 'hasPage' between the Web-page concept and other domain concepts. Although the generation of Web-page instances and the links between the Web-pages and domain concepts can be conducted by a new technique in an automated fashion, it still heavily relies on domain experts to build the ontology schema. This hinders the possibility of automating the knowledge discovery, representation and utilisation. In order to fulfil the goal of automating the processes of knowledge discovery, representation and application, this chapter addresses the main issue in semantic enhanced Web-page

recommendation: lack of general models for automated domain knowledge acquisition and domain ontology construction, which are applicable to any given websites. It presents a novel method for automatic construction of semantic knowledge of Web-pages which automatically processes from connecting domain terms together to forming a semantic network of Web-pages within a website, with the support of a new semantic knowledge representation model and a novel technique to automatically construct the knowledge base that covers domain terms, Web-pages and the relationships between them. This semantic network model can efficiently support semantic-enhanced Web-page recommender systems.

For the rest of the chapter, Section 5.2 presents preparation for semantic network construction. Section 5.3 proposes a new method of the automatic semantic network construction of Web-pages. Section 5.4 elaborates on the semantic network model. Section 5.5 evaluates experiment results. Section 5.6 gives some remarks of the proposed semantic network model. Finally, Section 5.7 summarizes this chapter.

5.2. Preparation for Semantic Network Construction

A semantic network of Web-pages refers to domain concepts and the relations between these concepts as well as Web-pages and the links between the domain concepts and Web-pages. The automatic approach to the semantic network construction of Web-pages aims at supporting automated knowledge discovery and knowledge representation in Web-page recommender systems.

The assumption used is the same as the one in Chapter 4, namely Web-pages are well designed and their titles carry key information about the page content. The Web-page titles are the metadata of Web-pages and they play an important role in expressing the topics of Web-pages as each title is often a phrase of keywords which distinguish Web-pages within a website. Therefore, the collection of Web-page titles of accessed Web-pages at the given website is chosen as the resource for domain knowledge discovery. The following factors have influenced this choice from a research perspective.

- It is a broad set of Web-page titles that was likely to yield a large number of extracted terms and association relationships between terms. These issues can be resolved by using the *in-out* relationship of terms in the semantic network which will be explained later.
- The titles of Web-pages usually include many insignificant words that may slow down data processing and decrease the quality of semantic network.
- Determination of linking relationships between domain terms and Web-pages is a very challenging task.

The domain terms are extracted from the collection of Web-page titles. Based on the co-occurrence relations between keywords in titles, we can develop a collocation map of domain terms of a given website, as a semantic network. The relations in this network are the linking relationships between domain terms for enabling a flexible and effective representation of domain terms (as collocations of terms) in the network, which are different from traditional relationships in normal ontologies. In addition, this semantic network is similar to the domain ontology structure which has two concept classes describe a Web-page and a domain term, and contains the association relations between Web-pages and respective terms in the network. This makes the semantic network as a knowledge representation model being machine-understandable, and supports reasoning about relations between Web-pages and domain terms in Web-page recommender systems.

Experimental evaluation will be carried out using the Web logs of a university website as the data source. The advantages of using a Web log as the data source of the semantic network are twofold: (i) it is easy and cheap to collect Web logs from the Web server, and (ii) by mining Web logs, we can obtain the accessed Web-pages, so extracted domain terms are more relevant to Web users' interests.

5.3. A New Method to Automatically Construct a Semantic Network of Web-pages

As stated in the previous section, the purpose of the automated construction of semantic network of Web-pages is to facilitate automated processes discovering and representing the semantic knowledge of visited Web-pages of a website for supporting more effective Web-page recommendations. In order to fulfil this purpose, a novel method is proposed to take Web logs of a given website as the input and to produce the semantic network of Web-pages for this website automatically. The flow diagram of implementing this method is shown in Figure 5-1, which consists of four processes: (1) accessed Web-page collection, (2) term extraction, (3) semantic network population of Web-pages, and (4) evaluation.

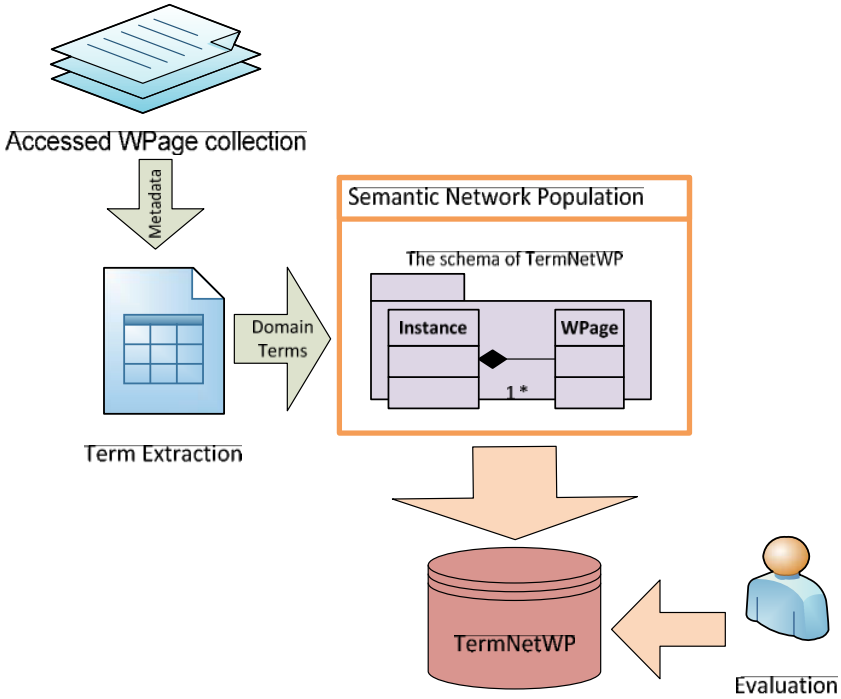


Figure 5-1: Flow diagram for automatic construction of a semantic network of Web-pages

***(1) Collection of Accessed Web-pages*:**

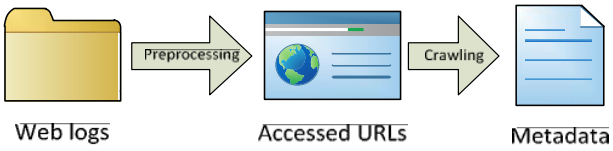


Figure 5-2: Process of collecting the accessed Web-pages

This process firstly pre-processes Web logs to extract the URLs of Web-pages that have been visited by users at the given website, and then the URLs are crawled to fetch the metadata of Web-pages, i.e. the titles of Web-pages based on the TITLE tag on the HTML documents of Web-pages.

(2) Extraction of Domain Terms:

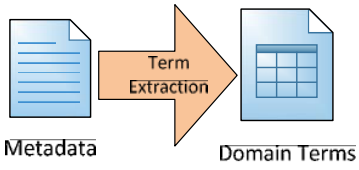


Figure 5-3: Process of extracting domain terms

This process extracts the domain terms from the titles of Web-pages retrieved in the first process (1). A term extraction algorithm is designed to extract terms from the Web-page titles. With this algorithm, tokens are firstly extracted, and then domain terms are generated based on these tokens. Some terms are single word terms, and some are composite terms which are formed by combining the tokens that are collocated together all the time. For examples, “Customer” and “Guides” forms a composite term “Customer_Guides”, because they always go together. The results of this process are domain term sequences, each of which is a list of terms in the order as they appear in the titles. This process is critical to the

successful extraction of useful term patterns from the titles of Web-pages. The detailed explanation of the term extraction algorithm will be presented in Section 5.4.

(3) Construction of a Semantic Network of Web-pages:

Based on the term sequences obtained from Process (2), a semantic knowledge representation model is built according to a collocation map (Park, Han & Choi 1995) and the Markov models (Borges & Levene 2005), in which occurrence weights of terms and associations between terms are taken into account to assess the frequencies of terms and collocations in the domain. Since the term sequences and Web-pages are associated with each other, both Web-pages and domain terms with their associations are included in this model. The schema of this model is designed to represent the domain terms, Web-pages, and the relationships between them. For a given website, this schema can be populated to form a semantic network of Web-pages, referred to as TermNetWP. This network is the domain knowledge base of this website. The details about how to construct such a domain knowledge base for a given website will be explained in Section 5.4.

(4) Evaluation

The last, but not the least, the evaluation process is for assessing the schema of TermNetWP, its population, and the effectiveness of the resulting semantic network. The evaluation measures will be presented in Section 5.5.

5.4. Modelling of Semantic Network of Web-pages

This section details the processes of term extraction, and semantic network construction of Web-pages. Furthermore, it presents how to reason about domain terms and Web-pages based on the populated semantic network. Finally, an experimental example of the semantic network construction is illustrated.

5.4.1. Term Extraction Algorithm

This algorithm, referred to as Algorithm 5-1, takes the Web-page titles obtained from Process (1) as the input and returns the domain terms of the Web-pages.

Firstly, it removes the stop words and/or invalid words in the titles and extracts tokens. Secondly, it identifies the domain terms by repeatedly checking all the tokens for forming composite terms until there are not any terms can be combined. Some tokens become the single word terms, while others become composite terms which consist of multiple tokens that often stay together and there is never any token sitting between these tokens. The domain terms are organized in the order as they appear in each title to become the domain term sequences as the output.

Algorithm 5-1: Term extraction

Input: *DT* (A set of titles)
Output: *TS* (A set of term sequences)
Process:
  Set *TS* = *null*
// *Token extraction*
  For each title in set *DT* {
    Remove invalid words, e.g. "&", ")"
    Remove stop words, e.g. "an", "and", "for"
    Split words in the title into a sequence of tokens, called X
      $X = t_1 t_2 \ldots t_n$; t_i ($i = [1..n]$): a token, n: the sequence length
    For each $\{t_i, t_{i+1}\}$ in X
      Add an *in-out* relationship between t_i and t_{i+1}
    Add X into *TS*
  }
// *Term combination*
  Repeat
    For each *in-out* relationship between two tokens t_i and t_j in *TS*
      If the occurrences of t_i and t_j are same
      then {
          combine t_i and t_j into one term $t_i\_t_j$
          remove t_i and t_j from *TS*
          add $t_i\_t_j$ into *TS*
          update the *in-out* relationships of $t_i\_t_j$ to others in *TS*
      }
  Until *Check-for-combination* == *false*
  Return *TS*

Procedure: ***Check-for-combination***
Description: Check if any pair of t_i and t_j always occurs concurrently in *TS*
Process:
  For each *in-out* relationship between two tokens t_i and t_j in *TS*
    If the occurrences of t_i and t_j are same
    Then return *true*
  Return *false*

For example, for the MS website, six domain term sequences are extracted from the titles of six Web-pages. These sequences are shown in Table 5-1.

Table 5-1: Sample of extracted domain term sequences

Page	Title	Domain term sequence
d_1	MS Word	MS, Word
d_2	MS Word Support	MS, Word, Support
d_3	MS Access	MS, Access
d_4	MS Access Support	MS, Access, Support
d_5	MS in Education	MS, Education
d_6	Visual Fox Pro Support	Visual, Fox_Pro, Support

5.4.2. Construction of Semantic Network of Web-pages

The semantic network of Web-pages, namely TermNetWP, is a network of domain terms extracted from the titles of visited Web-pages within the given website, the relationships between these terms, and the Web-pages that are mapped to these terms. This sub-section firstly defines the structure of TermNetWP, presents the schema of TermNetWP, and then explains how to populate TermNetWP.

1) Definitions

Definition 5.1. Let $T_{auto} = \{t_i : 1 \le i \le p\}$ be a set of domain terms, and $D = \{d_j : 1 \le j \le q\}$ be a set of Web-pages, each page d_j has a sequence of domain terms $X_j = t_1 t_2 \ldots t_n$, $t_k \in T_{auto}$ for $1 \le k \le n$ (a domain term may be duplicated in the sequence), which is extracted from the title of that page. Given a Web-page d_j and a domain term t_i, t_i is a domain term of d_j, as denoted as $t_i \tilde{\in} d_j$, if $t_i \in \{t_1, t_2, \ldots, t_n\}$.

For example, Table 5-1 shows a Web-page set $D = \{d_1, d_2, d_3, d_4, d_5, d_6\}$. A set of extracted domain terms are T_{auto} = {"MS", "Word", "Support", "Access", "Education", "Visual", "Fox_Pro"}. For a domain term sequence X_1 = ("MS", "Word") that is extracted from the title of page d_1, let t_1 = "MS" and t_2 = "Word", then $t_1 \tilde{\in} d_1$, and $t_2 \tilde{\in} d_1$, i.e., t_1 and t_2 are the domain terms of Web-page d_1.

Definition 5.2. Based on Definition 5.1, let $tf(t, d)$ be the number of occurrences of domain term $t \in T_{auto}$ in the title of Web-page $d \in D$, where $t \tilde{\in} d$. Given the Web-page set D and a domain term t, $tf(t, D) := \sum_{d \in D} tf(t, d)$ is defined as the number of occurrences of t over D.

For the example in Table 5-1, let term t_1 = "MS", which occurs 5 times in the domain term sequences, then $tf(t_1, D) = 5$. Let t_2 = "Word", which occurs 2 times in the domain term sequences, then have $tf(t_2, D) = 2$.

Definition 5.3. Based on Definition 5.1, let $TS = \{X_j : 1 \le j \le q\}$ be a set of domain term sequences, where X_j is a domain term sequence extracted from the title of page d_j; and $\omega_j(t_x, t_y)$ be the number of times that t_x is followed by t_y in X_j, i.e. $(t_x t_y) \subseteq X_j$, and there is no term between them. Given D and a pair of domain terms (t_x, t_y), $t_x, t_y \in T_{auto}$, $\omega(t_x, t_y) := \sum_{j=1}^{q} \omega_j(t_x, t_y)$ is defined as the number of times that t_x is followed by t_y in TS.

For the example in Table 5-1, let t_1 = "MS", t_2 = "Word", and t_3 = "Support". There are two times that term "Word" follows term "MS" in the domain term sequences, thus $\omega(t_1, t_2) = 2$. There is one time that term "Support" follows term "Word" in the domain term sequences, thus $\omega(t_2, t_3) = 1$.

Definition 5.4. **(Semantic network of Web-pages - TermNetWP)** By Definitions 5.1, 5.2 and 5.3, the semantic network of Web-pages, namely TermNetWP, is defined as a 4-tuples:

$$O_{auto} := <T, L, D, R>, \tag{6.1}$$

where

$T = \{(t, f) : t \in T_{auto} \wedge f = tf(t, D) > 0\}$ is a set of domain terms and the corresponding occurrences,

$L = \{(t_x, t_y, w_{xy}) : t_x, t_y \in T_{auto} \wedge (t_x t_y) \subseteq X_j \wedge X_j \in TS \wedge w_{xy} = \omega(t_x, t_y) > 0\}$ is a set of associations between t_x and t_y (with weight w_{xy}), and

$R = \{(t, d) : t \tilde{\in} d \wedge t \in T_{auto} \wedge d \in D\}$ is a set of relations between domain term t and Web-page d, that is, term t belongs to the title of page d.

For the example of MS website in Table 5-1, its TermNetWP consists of T = {("MS", 5), ("Word", 2), ("Support", 3), …}, L = {("MS", "Word", 2), ("Word", "Support", 1), …}, $D = \{d_1, d_2, ..., d_6\}$, and R = {("MS", d_1), ("Word", d_1), …}. According to Sowa (1991), TermNetWP can be represented graphically in patterns of interconnected concept nodes and arcs. The grapphical representation is shown in Figure 5-4.

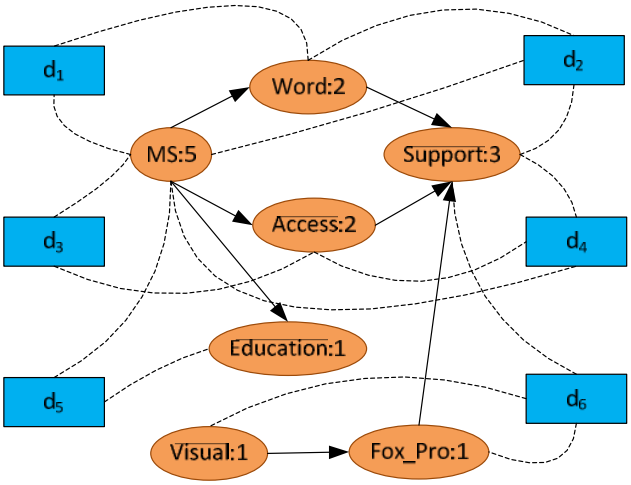


Figure 5-4: Graphical representation of TermNetWP

Ovals represent the domain terms along with their occurrences, and the rectangles represent Web-pages.

2) Schema of TermNetWP Model

Based on the definition of TermNetWP, its schema is designed to model domain terms, Web-pages and the relationships between them, as shown in Figure 5-5.

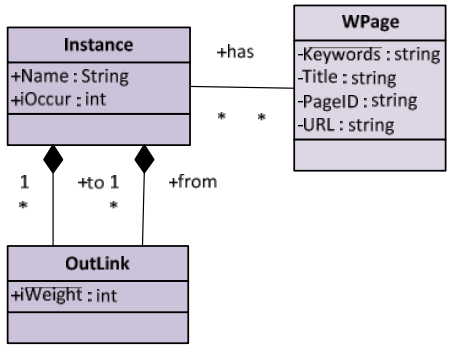


Figure 5-5: The schema of TermNetWP

Class *Instance* defines a domain term, i.e. $t \in T_{auto}$. Each term has two data type properties, which are *Name* and *iOccur*, and one *WPage* object property. The *iOccur* property refers to the number of occurrences of the term in the set of Web-page titles. Class *WPage* defines a Web-page, i.e. $d \in D$, with properties *Title*, *PageID*, *URL* and *Keywords* in the title. The *Keywords* property is used to store terms in a Web-page title. Classes *Instance* and *WPage* are associated through the 'hasWPage' relationship, i.e. $(t, d) \in R$, from *Instance* to *WPage*, which annotates a constraint: a term instance has one or some Web-pages; and the 'belongto-Instance' relationship, which is the inverse relationship of 'hasWPage', annotates a constraint: a Web-page belongs to one or more term instances.

In addition, class *OutLink* defines the *in-out* relationship between two terms to model the navigational relationship of terms in a term sequence. Class *OutLink* is used to connect *from* one term instance (t_x) to another term instance (t_y), and includes the corresponding connection weight ($iWeight = w_{xy}$). Class *OutLink* therefore has two object properties: (1) "from-Instance" referring to one term instance that is the first endpoint of a connection, and (2) "to-Instance" referring to one term instance that is the second endpoint of the connection. Correspondingly, class *Instance* also has two object properties: (1) 'hasOutLink' being the inverse of the 'from-Instance' relation, and (2) 'fromOutLink' being the inverse of the 'to-Instance' relation.

Different from traditional ontologies, the 'is-a' relationship is not used in TermNetWP, but the 'from-Instance', 'to-Instance', and 'hasWPage' relationships are used for supporting more efficiently reasoning about relevant domain terms and linked Web-pages.

3) Algorithm for constructing the semantic network of Web-Pages

Based on the designed schema of TermNetWP, an algorithm is proposed to automatically construct a semantic network of Web-pages, referred to as Algorithm 5-2, as shown in the following. The input data of the algorithm is a term sequence collection (*TSC*) in which each record consists of:

- The PageID of a Web-page $d \in D$;

- A sequence of terms $X = t_1 t_2 \ldots t_m \in TS$, $m > 0$, extracted from the title of the Web-page; and
- The URL of the Web-page.

Algorithm 5-2: Automatically constructing a TermNetWP

Input: *TSC* (Term sequence collection)
Output: *G* (TermNetWP)
Process:
Let *TSC* = {*PageID*, $X = t_1 t_2 \ldots t_m$, *URL*}
Initialize *G*
Let *R* = root or the start node of *G*
Let *E* = the end node of *G*
For each *PageID* and **each** sequence *X* in *TSC* {
Initialize a *WPage* object identified as *PageID*
For each term $t_i \in X$, $i = [1 .. m]$ {
If node t_i is not found in *G*, then
- Initialize an *Instance* object *I* as a node of *G*
- Set $I.Name = t_i$

Else
- Set *I* = the *Instance* object named t_i in *G*

Increase *I.iOccur* by 1
If ($i==0$) then
- Initialize an *OutLink* $R\text{-}t_i$ if not found
- Increase $R\text{-}t_i.iWeight$ by 1
- Set $R\text{-}t_i.fromInstance = R$
- Set $R\text{-}t_i.toInstance = I$

If ($i>0$) then
- Get *preI* = the *Instance* object with name t_{i-1}
- Initialize an *OutLink* $t_{i-1}\text{-}t_i$ if not found
- Increase $t_{i-1}\text{-}t_i.iWeight$ by 1
- Set $t_{i-1}\text{-}t_i.toInstance = I$
- Set $t_{i-1}\text{-}t_i.fromInstance = preI$

If ($i==n$) then
- Initialize an *OutLink* $t_i\text{-}E$ if not found
- Increase $t_i\text{-}E.iWeight$ by 1
- Set $t_i\text{-}E.toInstance = E$
- Set $t_i\text{-}E.fromInstance = I$

Set *I.hasWPage* = *PageID*
Add term t_i into *PageID.Keywords*
}
}

By using this schema, the TermNetWP for a given website can be developed to include the domain terms, navigational relationships between domain terms along with their weights, and the Web-pages mapped to respective domain terms. This schema can be implemented using the ontology language, e.g. OWL. The navigational relationships of domain terms can be represented by the 'from-Instance' and 'to-Instance' object properties.

For example, let $X = t_1 t_2 \ldots t_m$ be a term sequence, where m is the sequence length, and t_k ($1 \le k \le m$) is the k^{th} term in the sequence. By applying Algorithm 5-2, t_k is assigned to an *Instance* object, and t_{k-1}-t_k and t_k-t_{k+1} are assigned to *OutLink* objects. *OutLink* objects t_{k-1}-t_k and t_k-t_{k+1} point to *Instance* object t_{k-1} through the 'from-Instance' relation, and *Instance* object t_{k+1} through the "to-Instance" relation, respectively.

The semantic network of a given website, TermNetWP, can be used effectively not only to model the term sequences, but also to present the navigational relationships between terms in the term sequences based on the following features: (i) it allows a term node to have multiple in-links and/or out-links, so it has high level expressive power in describing the relationships among terms/nodes in the semantic network, i.e. one node might have previous or next nodes; and (ii) it includes the Web-pages whose titles contain the linked terms so that the meaning of Web-pages can be understood through these terms by software agents/systems.

An example

As an example, the Algorithm 5-2 is applied to construct a TermNetWP for the MS website. The given term sequences are extracted from Pages d_1-d_6 in Table 5-1. The constructed TermNetWP is depicted in Figure 5-4. Each d_j (j = [1..6]) is mapped to a number of terms, e.g. "MS", "Word", "Support", etc., which are the domain terms of page d_j. For example, domain term "MS" occurs five times in the title set of six pages d_1-d_6, so we have "MS:5" mapped to pages d_1-d_5. Since "MS" appears before "Word", "Access", and "Education" in the all titles, there are three navigation relations from "MS" to the three other domain terms.

5.4.3. Reasoning Algorithms for the Semantic Network of Web-pages

As mentioned before, TermNetWP represents the domain terms of Web-pages, Web-pages, and the associations among them. More importantly, the semantic network can be used for querying about the domain terms of a given page or Web-pages for a given domain term within a specific domain. The name of an instance is a domain term, so Web-pages whose title contains the term will be mapped to the corresponding term instance through the

'belongto-Instance' relation in the semantic network. On the other hand, domain terms included in a Web-page title can be specified by the *Keywords* property in class *WPage*. Therefore, the topic of a Web-page is able to be understood through either the domain terms of this page which can be retrieved from TermNetWP based on the links between terms and Web-pages, or the *Keywords* property values of this page.

This sub-section presents five reasoning algorithms based on TermNetWP for querying about:

(i) the domain terms of a given Web-page,
(ii) Web-pages mapped to a given domain term,
(iii) Web-pages mapped to a given set of domain terms,
(iv) Web-pages mapped to a given table of domain terms in which each domain term is assigned a prediction probability, and
(v) the domain terms of a given Web-page assigned a prediction probability.

These five reasoning algorithms will be used to support making Web-page recommendations in the Web-page recommender system of this research in Chapter 7.

Algorithm 5-3: Query about the domain terms of a given Web-page

This algorithm queries for the domain terms (topic) of a given Web-page $d \in D$, namely $\boldsymbol{Topic_{auto}(d)}$, by retrieving term instances that are associated with the *WPage* instance *d* via the 'belongto-Instance' object property (Algorithm 5-3). The returned domain terms are sorted in descending order of their occurrence weights based on the fact that the more times a domain term occurs on Web-pages, the more likely the term has been viewed. Based on Definitions 5.1-5.4, the algorithm is described in logics notation as follows.

$Topic_{auto}(d) = \{t_1, t_2, ..., t_s\}$,
where,
$d \in D = \{d_1, d_2, ..., d_q\}$,
$(t_i, d) \in \mathcal{R}$, $i = [1 .. s]$, and
$tf(t_i, D) > tf(t_j, D)$, $(i < j$ and $1 \le i, j \le s)$.

Algorithm 5-3: Query about domain terms of a given Web-page

Input:
d (PageID)
O (TermNetWP)
Output: T (list of domain terms)
Process:
Traverse through *O* to get the *WPage* instance whose PageID is *d*
Set *T* = term instances associated with Web-page d_i via the 'belongto-Instance' object property
Sort *T* in descending order of their occurrences
Return *T*

For instance, based on the sample TermNetWP illustrated in Figure 5-4, for a given Web-page, say d_2, Algorithm 5-3 can be used to query for the domain terms that are related to this Web-page (d_2) by retrieving the terms that are linked with this page as {"MS", "Word", "Support"}. Since the terms linked to this page have different occurrence weight values, as 5, 2, and 3, respectively, the domain terms are presented in descending order of the weight values, as {"MS", "Support", "Word"}.

Note that if the given page is not in the TermNetWP, its domain terms will be extracted from its title, and then this new page and its domain terms are added into the TermNetWP using Algorithm 5-2. The returned results will be the domain terms of this page that are sorted in descending order of their occurrence weights in the TermNetWP.

Algorithm 5-4: Query about pages mapped to a given domain term

This algorithm, as Algorithm 5-4, queries for the Web-pages of a given domain term *t*, named $\boldsymbol{Page_{auto}(t)}$, by retrieving *WPage* instances that are mapped to the term instance *t* via the 'hasWPage' object property. The returned pages are sorted in ascending order of connection weights between the Web-pages and domain term *t*. A *connection weight* between a Web-page $d \in D$ and domain term $t \in T_{auto}$ in TermNetWP *O* is defined as the total of links from/to domain term *t* to/from the domain terms of Web-page *d*.

Definition 5.5. **(Connection weight between a page and a domain term)** Based on Definitions 5.1-5.4, the connection weight between a Web-page $d \in D$ and a domain term $t \in T_{auto}$ is defined as follows:

$$\eta(d,t) = \sum_{k=1,t_k \tilde{\in} d}^{n} \omega(t_k,t) + \omega(t,t_k), \quad (5.2)$$

where,
$n = |\{t_k: t_k \tilde{\in} d\}|$ is the number of domain terms in the title of page d.

Based on Definitions 5.1-5.5, Algorithm 5-4 is described in logics notation as follows:

$Page_{auto}(t) = \{d_1, d_2, \ldots, d_s\}$,
where,
$(t, d_i) \in \mathbb{R}$, $i = [1 .. s]$, and
$\eta(d_i, t) < \eta(d_j, t)$, $(i < j$ and $1 \le i, j \le s)$.

Algorithm 5-4: Query about pages mapped to a given domain term

Input:
t (domain term)
O (TermNetWP)
Output: *DS* (a set of pages mapped to *t*)
Process:
Traverse through *O* to get the term instance whose name is *t*
Set *DS* = the set of *WPage* instances that *t* has
Sort *DS* in ascending order of weight between each page *d* in *DS* and *t* (using *getWeight*($d \in DS$, *t*, *O*))
Return *DS*

Procedure: *getWeight(Page, DomainTerm, TermNetWP)* – compute the connection weight between *Page* and *DomainTerm*
Set *ins* = term instances belonging to *Page* in *TermNetWP*
Set *W = 0*
For each *ins_i* in *ins*
Traverse through *TermNetWP* to find all links from *ins_i* to *DomainTerm* and from *DomainTerm* to *ins_i*
W += count of the links from *ins_i* to *DomainTerm*
+ count of the links from *DomainTerm* to *ins_i*
Return *W*

For instance, based on the sample TermNetWP illustrated in Figure 5-4, for a given domain term, say "Word", Algorithm 5-4 can be used to query for the Web-pages that are mapped to this domain term as $\{d_1, d_2\}$. The connection weights between domain term "Word" and these two pages in TermNetWP *O* can be calculated using *getWeight*(d_1, "Word", *O*) and *getWeight*(d_2, "Word", *O*), respectively. The connection weight for d_1 is 1, and for d_2 is 2. Therefore, the returned pages are presented in ascending order of the weight values, as $\{d_1, d_2\}$.

Algorithm 5-5: Query about Web-pages mapped to a given set of domain terms

When a page is accessed by a user, its domain terms are being viewed. Therefore, to recommend next Web-pages given the last accessed pages, the set of the domain terms of

the accessed pages should be taken into account rather than each domain term alone. This algorithm queries for Web-pages mapped to a given set τ of domain terms, i.e. $\tau \subset T_{auto}$, named $\boldsymbol{Page_{auto}(\tau)}$, by retrieving Web-pages associated with each domain term in the given set via the 'hasWPage' object property (Algorithm 5-5). In order to ensure that the degree of relevance of the pages to domain term set τ is taken into account in the Web-page recommendation process later, the returned pages are sorted in descending order of their *correlation* weights. The correlation weight between a Web-page and a domain term set is defined as follows.

Definition 5.6. **(Correlation weight between a Web-page and a domain term set).** Based on Definitions 5.1-5.4, the correlation weight between a Web-page $d_i \in D$ and a set of domain terms $\tau \subset T_{auto}$ is defined as the number of domain terms in τ which appear in the title of Web-page d_i,

$$wpt\_f(d_i, \tau) = |\{t_k : t_k \tilde{\in} d_i\} \cap \tau|. \quad (5.3)$$

Based on Definitions 5.1-5.4 and 5.6, Algorithm 5-5 is described in logics notation as follows:

$Page_{auto}(\tau) = \{d_1, d_2, \ldots, d_s\}$,
where,
$(t, d_i) \in \mathbb{R}$, $t \in \tau$, $i = [1 .. s]$, and
$wpt\_f(d_i, \tau) > wpt\_f(d_j, \tau)$, $(i < j$ and $1 \le i, j \le s)$.

Algorithm 5-5: Query about pages mapped to a given set of domain terms

Input:
  τ (set of domain terms)
  O (TermNetWP)
Output: DS (set of pages mapped to τ)
Process:
  Set DS = null
  For each domain term t in τ
    Traverse through O to get term instance whose name is t
    Set D_i = the *WPage* instances t has
    The correlation weight of each *WPage* instance and τ is counted every time the page is found in D_i
    Add D_i into DS
  Sort DS in descending order of the counted weights
  Return DS

For instance, based on the sample TermNetWP illustrated in Figure 5-4, for a given domain term set, as τ = {"MS", "Word", "Support"}, Algorithm 5-5 can be used to retrieve the Web-pages that are mapped to this domain term set as $DS = \{d_1, d_2, d_3, d_4, d_5, d_6\}$. Since the correlation weights between each page in *DS* and τ are 2, 3, 1, 2, 1, and 1, respectively, the resulting pages are presented in descending order of the weight values, as $\{d_2, d_1, d_4, d_3, d_5, d_6\}$. It is shown that the pages which are more closely related to the given domain terms, e.g. d_2, d_1, d_4, are listed first.

Algorithm 5-6: Query about Web-pages mapped to a given set of domain terms, in which each domain term is assigned a prediction probability

In the Web-page recommendation context, it is common to predict the potential next viewed domain terms given the domain terms that are recently viewed with estimated probability values and then to recommend Web-pages that are mapped to these predicted domain terms. This algorithm is designed for this purpose. It queries for Web-pages that are mapped to the next viewed domain terms with the prediction probabilities of the predicted domain terms taken into account. It finds the mapped Web-pages by retrieving Web-pages associated with each domain term in the given domain term set via the 'hasWPage' object property, named ***PageProb(τ')*** (Algorithm 5-6), given the set of domain terms along with respective prediction probabilities (τ'). In order to ensure that Web-pages which are mapped to the domain terms with higher prediction probabilities are listed first, the returned pages are sorted in descending order of their *correlation proportions* with the predicted domain term set τ'. Formally, based on Definitions 5.1-5.4, the *correlation proportion* of a predicted page which is mapped to a set of predicted domain terms is defined, as follows:

Definition 5.7 **(Correlation proportion).**

Based on Definitions 5.1-5.4, and given

τ' = {(t, ρ)} as a set of domain terms along with respective prediction probabilities, where $t \in T_{auto}$, ρ: the prediction probability of t, ρ = [0, 1], and

a Web-page $d_i \in D$, whose title contains a domain term t in τ'.

Let $\tau'_i = \{(t_k, \rho_k) : t_k \tilde{\in} d_i \wedge (t_k, \rho_k) \in \tau'\}$, and $M = |\tau'_i|$.

The *correlation proportion* of d_i and τ' can be defined, as follows:

$$wpt\_f'(d_i, \tau') = \sum_{k=1}^{M} \frac{\rho_k}{|\{t_i: t_i \tilde{\in} d_i\}|}. \quad (5.4)$$

Based on Definitions 5.1-5.4 and 5.7, Algorithm 5-6 is described in logics notation as follows:

PageProb(τ') = {d_1, d_2, …, d_s},
where,
τ' = {(t, ρ)},
(t, d_i) ∈ $\mathbb{R}$, i = [1 .. s], and
$wpt\_f'(d_i, \tau') > wpt\_f'(d_j, \tau')$, ($i < j$ and $1 \le i, j \le s$).

Algorithm 5-6: Query about Web-pages mapped to a given set of domain terms, in which each domain term is assigned a prediction probability

Input:
    τ'({domain terms, probs})
    O (TermNetWP)
Output: DS – a set of pages mapped to τ'
Process:
    Set DS = null
    For each (domain term t and its probability ρ) in τ'
        Traverse through O to get the term instance whose name is t
        Set DS_i = the set of *WPage* instances which t has
        For each d in DS_i
            Set τ_i = term instances belonging to d
            Update the correlation proportion of page d, as follows:
                If $d \in DS$ then
                    $d.\omega = d.\omega + \rho / \tau_i$.size()
                Else
                    $d.\omega = \rho / \tau_i$.size()
                    Add (d, $d.\omega$) into DS
    Sort DS in descending order of the correlation proportions ω
    Return DS

For instance, for the MS website, whose TermNetWP is illustrated in Figure 5-4, given a set of predicted domain terms with respective estimated probabilities as τ' = {("MS":0.5), ("Word":0.3), ("Support":0.2)}, where the summation of all probability values can be different from 1, Algorithm 5-6 can query for the Web-pages which are mapped to this domain term set as {d_1, d_2, d_3, d_4, d_5, d_6}. Since the correlation proportions between these pages and domain term set τ' are calculated, as follows:

$$wpt'_f(d_1, \tau') = \frac{0.3}{2} + \frac{0.5}{2} = 0.4,\ wpt'_f(d_2, \tau') = \frac{0.5}{3} + \frac{0.3}{3} + \frac{0.2}{3} = 0.333,$$

$$wpt'_f(d_3, \tau') = \frac{0.5}{2} = 0.25,\ wpt'_f(d_4, \tau') = \frac{0.5}{3} + \frac{0.2}{3} = 0.233,$$

$wpt'_f(d_5, \tau') = \frac{0.5}{2} = 0.25$, and $wpt'_f(d_6, \tau') = \frac{0.2}{3} = 0.067$,

the pages are presented in descending order of the correlation proportions, as $\{d_1, d_2, d_3, d_5, d_4, d_6\}$.

It can be seen that the results are presented in a different order compared with the results from the ones obtained from Algorithm 5-5, although the returned Web-pages are the same. This indicates that the domain term prediction probabilities influence the order of predicted Web-pages.

Algorithm 5-7: Query about the domain terms of a given Web-page assigned a prediction probability

This algorithm is for the situation that the domain terms which are mapped to the predicted next pages are needed. It queries for the domain terms of a given Web-page $d \in D$ with a prediction probability φ, named ***Topic(d, φ)***. This algorithm retrieves the term instances that are associated with the Web-page, and each returned term has the same prediction probability as that of the Web-page, as shown in Algorithm 5-7. Based on Definitions 5.1-5.4, the algorithm is described in logics notation as $Topic(d, \varphi) = \{(t, \rho) : (t, d) \in \mathbb{R} \wedge \rho = \varphi\}$.

Algorithm 5-7: Query about domain terms of a given Web-page assigned a prediction probability

Input:
  d_i, φ (PageID, Probability)
  O (DO-WP)
Output: T (set of domain terms & φ)
Process:
  Traverse through O to get *WPage* instance whose PageID is d_i
  Set T = term instances associated to Web-page d_i
    In T, φ is assigned to each term
  Return T

5.4.4. An Experimental Example of Semantic Network Construction

This sub-section illustrates an example of constructing the semantic network for a real-world website, the handbook website of the University of Technology, Sydney (UTS) (www.handbook.uts.edu.au/). Referring to the flow diagram shown in Figure 5-1, the input

is the Web log file of this website. Firstly, a set of accessed Web-pages are acquired after pre-processing the Web log data. Secondly, titles are retrieved from the Web-pages. Thirdly, a set of domain term sequences of the Web-pages are extracted from the titles using Algorithm 5-1. Finally, a TermNetWP is automatically generated given the domain term sequences using Algorithm 5-2. An ontology development tool Protégé is used to design the schema of TermNetWP and to generate Java schema for processing TermNetWP population.

Figure 5-6 illustrates the TermNetWP for a small sample of the input data for the UTS Handbook website. The inputs include the PageIDs, titles and URLs of Web-pages. Four sample pages are presented as shown in the top part of Figure 5-6.

Given the four Web-page titles, the tokens/simple terms extracted by the term extraction algorithm are: *STM90562*, *STM90528*, *STM90284*, *STM90285*, *Core subjects*, *Accounting*, *Biotechnology*, *Certificate Entry EN*, and *Adult Education. Core subjects, Certificate Entry EN*, and *Adult Education* are the composite terms. The TermNetWP is then constructed based on the schema of TermNetWP and these terms. Each term becomes an *Instance* object. Instances are connected to each other by *OutLinks* via the 'from-Instance' and 'to-Instance' relations. The TermNetWP is shown in Figure 5-6 (the bottom part). For instance, it can be seen from the TermNetWP, instance *Core_subjects* has previous instances *STM90562*, *STM90528*, *STM90284*, and *STM90285*, which occur before, and next instances *Accounting*, *Biotechnology*, *Certificate_Entry_EN*, and *Adult_Education*, which occur after. The following instances are *UTS* and *Handbook*. The *WPage* instances *10582*, *10583*, *10589*, and *10588* are mapped to related instances; these relations are not shown in the figure. Instances *UTS* and *STM90562*, *STM90528*, *STM90284*, and *STM90285* are connected via *OutLinks UTS_STM90562*, *UTS_ STM90528*, *UTS_ STM90284*, and *UTS_ STM90285*, respectively.

Based on this TermNetWP, we can query for all Web-pages which are mapped to some terms, and vice versa. For example, as in Figure 5-6, if we query for concepts which are associated after *core_subjects*, then the system will return *Accounting, Biotechnology, Certificate Entry EN*, and *Adult Education*. These results show the reasonable collocations

of the domain terms in the specific domain. On the other hand, we also can query for Web-pages which are mapped to *core_subjects*, and results are *10583*, *10582*, *10589*, and *10588*.

In addition, the semantic network, TermNetWP, enables to infer the related pages using the keywords of a Web-page. For example, when a user is visiting Web-page *10582* which has keywords *UTS*, *STM90285*, *Core subjects*, *Accounting*, and *Handbook* in its title, the system can find out the related Web-pages *10583*, *10589*, and *10588*, which are mapped to *Handbook* and *Core subjects*. These reasoning results show that the constructed TernNetWP is correct and meaningful.

The semantic network of a website, namely TermNetWP, will be verified using some evaluation measures in the following section. It will be further validated when it is applied to the Web-page recommender system in Chapter 7.

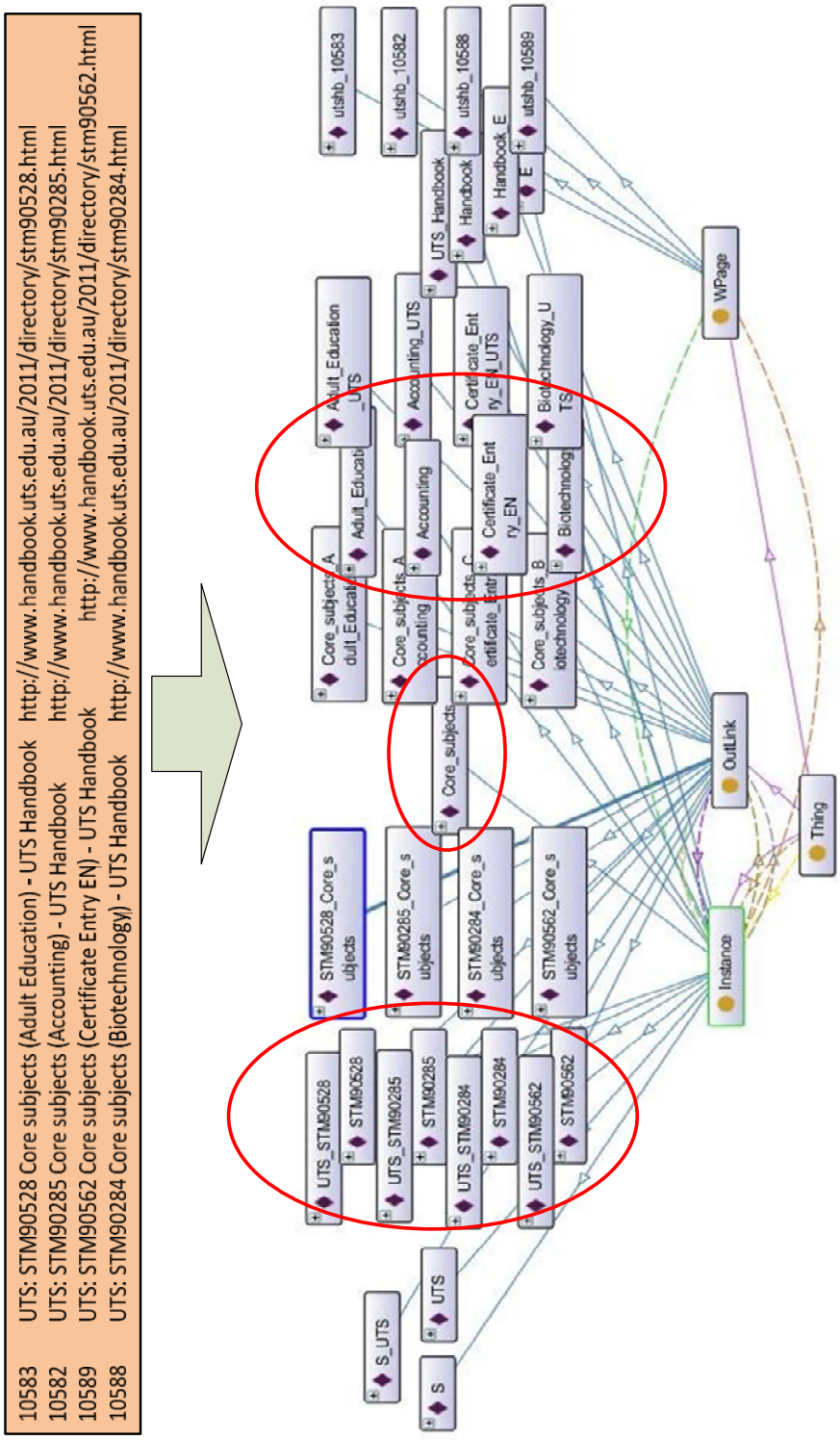


Figure 5-6: TermNetWP population: A set of titles => the corresponding term instances and relations in the semantic network of Web-pages

5.5. Experimental Evaluation

This section firstly presents evaluation measures, and then presents the experiment results of the TermNetWP model for a given website.

5.5.1. Evaluation Measures

Evaluation of an automatically constructed semantic network is a delicate issue as it is not clear what one could compare to. It is similar to the issue of ontology learning. To address this issue, some approaches are to approximate the appropriateness of some ontologies by means, such as the "corpus fit" or consistency of an ontology (Cimiano et al. 2009). Some others (Tartir et al. 2005) have presented ontology metrics to evaluate the quality of ontology construction, referred to as the first evaluation method. On the other hand, Chen & Chuang (2008) used *concept precision* (CP) and *concept location precision* (CLP) to evaluate the effectiveness of their automatically constructed ontology, referred to as the second evaluation method. Since TermNetWP is constructed in ontology style, this section uses both evaluation methods to evaluate the TermNetWP construction. With the first evaluation method, the metrics discussed in Section 2.3.9 are used. With the second evaluation method, CP and CLP are considered in a compatible manner with the case of TermNetWP. In the context of the semantic network evaluation, the generated terms and their relations are taken into account. With these evaluation methods, the domain terms populated in the TermNetWP are evaluated by comparing them with the collocations of terms in the titles. A term is considered to be accepted if it satisfies the combination condition of Algorithm 5-1. Location of a term in the generated TermNetWP is considered to be right if it meets the position of the term in titles, and is properly mapped to Web-pages. Thus, the two measures of *precision* evaluation can be calculated by Equations (2.6) and (2.7), introduced in Section 2.3.9, where A is the number of terms accepted; B is the number of terms not accepted; C is the number of terms whose locations are right; and D is the number of terms whose locations are wrong.

5.5.2. Experiment Results

An experiment was conducted to evaluate the TermNetWP of UTS Handbook website using the two evaluation methods. The data sources of the experiments are the Web server log of the UTS handbook website. TermNetWP population is implemented using the schema of TermNetWP following the steps presented in Section 5.3. In order to easily evaluate the TermNetWP construction using the two mentioned evaluation methods, 1000 Web-page titles of the given website were selected for TermNetWP population.

Regarding the first evaluation method, the schema and instance metrics are measured. The total of instances 4910 are distributed across all the defined classes *Instance*, *OutLink* and *WPage*, so the Class Richness is 1. Class Instance Distribution is 1001.451. Class *Instance* is connected to other classes via the 'from-Instance', 'to-Instance', and 'hasWPage' relationships which also have the correspondingly inverse relationships. Hence, the Class Connectivity of *Instance* equals the total number of the 'from-Instance', 'to-Instance', and 'hasWPage' relationship instances. Correspondingly, the number of the 'from-Instance' and 'to-Instance' relationship instances is the Class Connectivity of *OutLink*; and the number of the 'hasWPage' relationship instances is the Class Connectivity of *WPage*. Consequently, the results of measurements for each class are shown in Table 5-2.

Table 5-2: TermNetWP evaluation

Class	# Instances	Connectivity	Importance	RU
Instance	1119	12392	0.228	1
OutLink	2791	5582	0.568	1
WPage	1000	6810	0.204	1

It can be seen from Table 5-2 that the importance values of two classes *Instance* and *WPage* are similar, but less than the value for class *OutLink* which is most important to present the relationships between term instances in the domain. Relationship Utilizations (RU) show that all the defined relationships are used at the instances level. Generally speaking, the schema of TermNetWP and the populated TermNetWP are correctly implemented.

Regarding the second evaluation method, CP and CLP are considered. All the generated terms have been verified, and it has been found that the most of terms are distributed and collocated precisely in the network. The CP and the CLP of the TernNetWP are nearly 100%, which reflects the term extraction algorithm was applied effectively to extract the acceptable terms; the collocations of almost all the terms are accepted, and almost all the terms are properly mapped to Web-pages.

Although there are no other models to be compared with the proposed model, the precision results show that the accuracy of the model is acceptable (nearly 100%). However, the CP and the CLP are able to be affected by the content of titles. For example, if there are some special characters or invalid words in some titles, the system will have trouble with parsing these titles. This may cause some errors when extracting tokens/simple terms for the term sequences from these titles. However, these errors can be overcome by pre-processing the Web-pages to filter the troublesome words. Overall, the proposed method of automatically construct the semantic network of a website is effective to extract domain terms given the Web logs and to construct the semantic network of Web-pages (TermNetWP) based on those terms.

5.6. Remarks

The strength and limitation of the method to automatically construct a semantic network of Web-pages and its applications are summarized as follows:

- Strengths:
  - Since the proposed schema of TermNetWP takes the advantages of existing concept learning models, such as the Markov model, and the collocation map, it can represent domain terms and Web-pages in a self-contained and compact semantic network.
  - This method can automatically construct a semantic network of a website. It reduces significantly the effort in constructing a domain ontology of a website for Web-page recommendation.

  - The semantic network of Web-pages (TermNetWP) can support the reasoning about the relationships between terms and Web-pages, i.e., it enables reasoning about the relevant domain terms of a Web-page and the relevant Web-pages of one or more domain terms.
  - The ontological implementation of TermNetWP enables a smooth integration of this network with a semantic-enhanced Web-page recommendation process, since it enables the semantic information of the Web-pages to be machine-understandable.
  - Since domain terms are associated based on their co-occurence relations in titles, this enables to query for relevant domain terms of a given domain term within the domain.
- Limitation:
  - Domain terms are extracted from Web-page titles.
  - Terms are not normalized and clustered in synonymous groups.
- Applications:
  - The semantic network of a website can be used for sharing the semantic knowledge of Web-pages in relevant Web applications, especially Web-page recommender systems.
  - The method can be used to automatically generate a semantic network of any website.
- The semantic network construction method can be combined with the Web content analysis for enriching the semantic knowledge representation of Web-pages.

5.7. Summary

In summary, this chapter has shown that it is possible to automatically generate a semantic network of Web-pages of a website. Given Web usage data, the proposed system can process data and automatically generate a semantic network of Web-pages, i.e. TermNetWP. The proposed term extraction algorithm has been applied to obtain domain terms for TermNetWP construction. In the experiments, most of populated terms and relationships are acceptable. The built TermNetWP is a self-contain network keeping

domain terms in order they appear in the Web-page titles, and domain terms are associated with respective Web-pages. It is useful for reasoning about semantic information relevant to domain terms and Web-pages.

With the help of TermNetWP, the system can query for the domain terms of a Web-page, or any Web-pages which are connected directly or indirectly to given domain terms in order to support Web-page recommendation. The directly connected Web-pages are more likely recommended as the relevance degree of the Web-pages and domain terms is higher. In Chapter 7, the effectiveness of TermNetWP will be validated by applying it to Web-page recommendation in the semantic-enhanced Web-page recommender system.

Chapter 6.

CONCEPT NAVIGATION MODELS FOR PREDICTION

6.1. Introduction

Addressing the objective 3 in Chapter 1, this chapter presents a new approach to Web usage knowledge representation, which enables the smooth integration of Web usage and domain knowledge of a given website. Two new concept navigation models are proposed to automatically generate a weighted network of concept navigation. The concept refers to Web-pages or domain terms of Web-pages. The first model uses the discovered Web usage knowledge, which is a collection of FWAP, short for frequent Web access patterns, to automatically generate a weighted network of Web-page navigation for a website, namely WPNavNet. The second model uses the discovered Web usage knowledge, i.e. FWAP, integrated with the domain knowledge, i.e. DomainOntoWP or TermNetWP, to automatically generate a weighted network of domain term navigation for a website, namely TermNavNet. Based on these two models, a number of reasoning algorithms are proposed to facilitate Web-page and domain term prediction for supporting effective Web-page recommendations in the next chapter.

This paper is structured as follows: Section 6.2 proposes a Web-page navigation model to automatically generate a WPNavNet, and presents reasoning algorithms on the model. Section 6.3 proposes a domain term navigation model to automatically generate a TermNavNet, and presents reasoning algorithms on the model. Section 6.4 presents an example of the proposed models. Section 6.5 summarizes this chapter.

6.2. A Web Usage Knowledge Representation Model of a Website for Web-page Recommendation

This section proposes the Web-page navigation model which is used to automatically generate a weighted network of frequent Web access patterns in ontology style, then describes the schema of the model, how to automatically construct the network, and the reasoning algorithms based on this model.

6.2.1. A Web-page Navigation Model

The Web-page navigation model is motivated by the Markov models, which is a kind of well-known probabilistic model. They have remarkable features of events prediction by learning sequential data patterns. The idea of Markov models is very useful for supporting Web-page recommendation.

According to (Borges & Levene 2004), the Markov models are efficient to model a collection of navigation records, and to estimate the transition probabilities between elements among the records. There are many descriptions of Markov models as in (Borges & Levene 2004; Grinstead & Snell 1997; Lam & Riedl 2004). In this study, the Markov models corresponding to the Hypertext Probability Grammar models (Borges & Levene 2004; Lam & Riedl 2004) are considered in developing the Web-page navigation model to predict next Web-pages for a given sequence of Web-pages.

In many applications, the first-order Markov models are often not very accurate in predicting the next page taken by a user, because these models do not look far enough in the past actions to discriminate different behavioural sequences (Deshpande & Karypis 2004). Therefore, the higher-order models, e.g. second or third order, are usually used to gain higher prediction accuracy. Unfortunately, the drawbacks of those higher-order models are high state-space complexity, reduced coverage (Liu, Mobasher & Nasraoui 2011), and sometimes low prediction accuracy (Deshpande & Karypis 2004). Borges & Levene (2004) proposed a dynamic clustering-based Markov model which extends the first-order model to obtain second-order transition probabilities. This model can increase prediction accuracy,

while still keeping run-time and memory space increasing less than building the normal second-order probability models.

Thanks to the advantages of Markov models of navigation sequence representation, the first and second-order Web-page navigation models are developed based on the first-order and second-order Markov models, respectively, to model the Web usage knowledge, i.e. FWAP, for the Web-page recommender system in this study. It is simple to develop the first-order model, but it is challenging to develop the second-one. Therefore, the dynamic clustering-based Markov model (Borges & Levene 2004) is adopted in developing the second-order Web-page navigation model.

The inputs to these models are the set of FWAP. The benefit of using FWAP instead of all Web access sequences is that a number of uninteresting Web-pages are removed, so a huge amount of unnecessary states each of which represents a Web-page is removed from the first and second-order Web-page navigation models.

The formal definitions that are used to describe the models are given as follows:

Definition 6.1 Let $D = \{d_j : 1 \le j \le q\}$ be a set of Web-pages, where q is the number of Web-pages, and $\mathbb{P} = \{P_1, P_2, \ldots, P_n\}$ be a set of FWAP, where each pattern P_i ($i = [1..n]$) contains a sequence of frequently visited Web-pages, n is the number of the patterns, and $P_i = d_{i1}d_{i2} \ldots d_{im}$, $d_{ij} \in D, j = [1..m]$, where m is the number of Web-pages in the pattern.

Definition 6.2 Given the Web-page pattern set $\mathbb{P}$ and a Web-page $d_x \in D$, we define $\delta_x = |\{P : P \in \mathbb{P} \wedge (d_x) \subseteq P\}|$ as the number of occurrences of d_x in $\mathbb{P}$. Given the Web-page pattern set $\mathbb{P}$ and a pair of pages (d_x, d_y), $d_x, d_y \in D$, we define $\delta_{x,y} = |\{P : P \in \mathbb{P} \wedge (d_x d_y) \subseteq P\}|$ as the number of times that d_x is followed by d_y in $\mathbb{P}$ and there is no page between them. We also define $\delta_{S,x}$ as the number of times page $d_x \in D$ is the first page in a Web-page pattern $P \in \mathbb{P}$, and $\delta_{x,E}$ as the number of times a Web-page pattern $P \in \mathbb{P}$ terminates at page $d_x \in D$.

Based on Definitions 6.1 and 6.2, the Web-page navigation model is proposed to automatically generate a weighted semantic network of frequently visited pages with the

weight being the probability of the transition between two adjacent pages based on FWAP. We refer to this Web-page navigation network as WPNavNet hereafter. Similar to the Markov model (Borges & Levene 2004), the Web-page navigation model is a self-contained and compact model. It has two main kinds of elements: (1) state nodes; (2) relations between state nodes. One state node presents the current state, e.g. the page currently visited by a user (or the current page for short), and may have some previous state nodes, e.g. pages previously visited by a user (or previous pages for short), and some next state nodes, e.g. next pages requested by a user (or next pages for short). By scanning each Web-page pattern $P_i \in \mathcal{P}$, each page becomes a state in the navigation model. There are also two additional states: a start state, *S*, representing the first state of every Web-page pattern; and a final state, *E*, representing the last state of every Web-page pattern. There is a transition corresponding to each pair of pages in a pattern, a transition from the start state *S* to the first page of a Web-page pattern, and a transition from the last page of a Web-page pattern to the final state *E*. The model is incrementally built by processing the complete collection of Web-page patterns.

The probability of a transition is estimated by the ratio of the number of times the corresponding sequence of states was traversed and the number of times the anchor state occurred. In the Web-page navigation model, the first-order and second-order transition probabilities are taken into account for Web-page prediction.

Definition 6.3 **(First-order prediction probability)** By Definitions 6.1 and 6.2, the first-order transition probabilities are estimated according to the following expressions:

$$\varphi_{S,x} = \frac{\delta_{S,x}}{\sum_{y=1}^{n} \delta_{S,y}}, \tag{6.1}$$

which is the first-order probability of the transition from the start state *S* to state d_x,

$$\varphi_{x,y} = \frac{\delta_{x,y}}{\delta_x}, \tag{6.2}$$

which is the first-order transition probability from state d_x to d_y,

$$\varphi_{x,E} = \frac{\delta_{x,E}}{\delta_x}, \tag{6.3}$$

which is the first-order transition probability from state d_x to the final state *E*.

Definition 6.4 **(Second-order transition probability)** According to Definitions 6.1 and 6.2, let $\varphi_{x,y,z}$ be the second-order transition probability, that is, the probability of the transition (d_y, d_z) given that the previous transition (d_x, d_y), where d_x, d_y, $d_z \in D$. The second-order probability can be estimated as follows:

$$\varphi_{x,y,z} = \frac{\delta_{x,y,z}}{\delta_{x,y}}, \tag{6.4}$$

where $\delta_{x,y} = |\{P : P \in \mathbb{P} \wedge (d_x d_y) \subseteq P\}|$, and $\delta_{x,y,z} = |\{P : P \in \mathbb{P} \wedge (d_x d_y d_z) \subseteq P\}|$.

Definition 6.5 **(Web-page navigation model)** Based on the above Definitions 6.1-6.4, the Web-page navigation mode is formally defined as a triple:

$$O_D := <\mathbb{N}_D, \boldsymbol{\Phi}_D, \mathbb{M}_D>, \tag{6.5}$$

where

$\mathbb{N}_D = \{(d_x, \delta_x) : d_x \in D\}$: a set of Web-pages along with the corresponding occurrence counts,

$\boldsymbol{\Phi}_D = \{(d_x, d_y, \delta_{x,y}, \varphi_{x,y}) : d_x, d_y \in D\}$: a set of transitions from d_x to d_y, along with their transition weights ($\delta_{x,y}$), and first-order transition probabilities ($\varphi_{x,y}$),

$\mathbb{M}_D = \{(d_x, d_y, d_z, \delta_{x,y,z}, \varphi_{x,y,z}) : d_x, d_y, d_z \in D\}$: a set of transitions from d_x, d_y to d_z, along with their transition weights ($\delta_{x,y,z}$), and second-order transition probabilities ($\varphi_{x,y,z}$).

If $\mathbb{M}_D$ is non-empty, the model (6.5) is considered as *the second-order Web-page navigation model*, otherwise *the first-order Web-page navigation model*.

An example of the first-order Web-page navigation model

Suppose that there is a set of WAP (short for Web access patterns) discovered from a website, as shown in Table 6-1,

Table 6-1: A set of Web access patterns

Pattern	NOP
d_1 d_2 d_3	3
d_1 d_2 d_4	1
d_5 d_2 d_4	3
d_5 d_2 d_6	1

where d_1 to d_6 are the Web-pages that this website has, a pattern is a sequence of Web-pages which are frequently visited by users at the given website and NOP refers to the number of occurrences of each pattern in the given WAP set.

Using the *first-order* Web-page navigation model, the WAP can be represented as follows: Each Web-page becomes a state. All the states are connected through the transition links. One state is connected to another state if the corresponding Web-pages are adjacent in at least one given pattern. For each pattern, a number of transitions are presented in the model, for example, transition (d_1, d_2) states that a user visited d_2 after d_1, transition (S, d_1) states that the user visited d_1 first, and transition (d_3, E) states that d_3 is the last page the user visited in the session.

For each transition from state d_i to d_j ($1 \leq i, j \leq 6$), we assign $\delta_{ij}(\varphi_{ij})$ with δ_{ij} representing the transition weight which is the number of times that users visit page d_j from page d_i and φ_{ij} representing an estimated first-order transition probability of the transition from d_j to d_i. Figure 6-1 shows the first-order Web-page navigation model representing a set of states with respect to the set of WAP in Table 6-1.

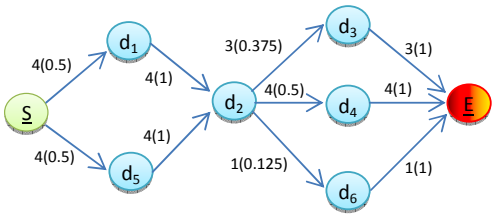


Figure 6-1: The first-order Web-page navigation model with respect to the patterns given in Table 6-1

6.2.2. Schema of Web-page Navigation Model

In order to automatically construct a WPNavNet for a given FWAP in the formal ontology language OWL, the schema of Web-page navigation model is designed like an ontology schema. The schema consists of classes *cNode* and *cOutLink*, and relationships between them, namely *inLink*, *outLink*, and *linkTo*, as shown in Figure 6-2, where *cNode* and *cOutLink* define the current state node and the association from the current state node to a next state node, respectively. Each state node represents a Web-page in FWAP.

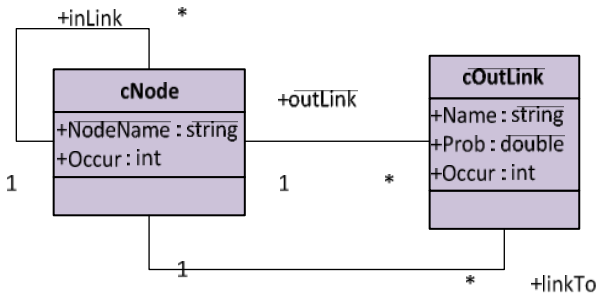


Figure 6-2: The schema of Web-page navigation model

Class *cNode* represents a page in the Web-page navigation model, and its page ID is assigned to the *NodeName* property value. Class *cNode* has two *inLink* and *outLink* object properties referring to *cNode* and *cOutLink*, respectively. The number of occurrences of the page is represented by *Occur*, i.e. δ_x. *inLink* represents an association from a previous state node, e.g. a previously visited page, to the state node it belongs to. Class *cOutLink* represents an association from a state node to another next state node with a transition probability *Prob*, e.g. $\varphi_{x,y}$, and the number of the associations is captured by its *Occur* property. Class *cOutLink* has a *linkTo* object property referring to a *cNode* object, i.e. the next state node, so the name of this *cNode* object is assigned to the *Name* property of *cOutLink*.

6.2.3. Automatic Construction of Web-page Navigation Model

The Web-page navigation model of a website, i.e. WPNavNet, can be automatically constructed by populating the schema of Web-page navigation model with a given set of FWAP for this website. The automatic construction process is presented in Algorithm 6-1.

Algorithm 6-1: WPNavNet construction

Input: $\mathcal{P}(FWAP)$
Output: M (WPNavNet)
Process:
Initialize M
For each $P = (d_1...d_m) \in \mathcal{P}$
For each $d_i \in P$, $i = [1..m]$
Initialize *cNode* objects with $NodeName = d_i$, d_{i-1}, d_{i+1} and $Occur = 1$ if they are not found in M
Initialize a *cOutLink* object with $Name = d_i\_d_{i+1}$ and $Occur = 1$ if it is not found in M
Increase $d_i.Occur$ and $d_i\_d_{i+1}.Occur$ if they are found in M
$d_i\_d_{i+1}.linkTo = d_{i+1}$
$d_i.outLink = d_i\_d_{i+1}$
$d_i.inLink = d_{i-1}$
Update all objects into M
Update transition probabilities in the *cOutLink* objects
Return M

The transition probabilities in the *cOutLink* objects can be updated based on the first-order or second-order probability formulae, i.e. (6.1-6.3) or (6.4), depending on which order of Web-page navigation model is applied. The second-order Web-page navigation model is more complicate than the first-order one because of the second-order transition probability computation. This second-order navigation model can be obtained by using the dynamic clustering-based Markov model (Borges & Levene 2004). According to this method, state nodes might be duplicated with respect to in-links whose corresponding second-order probabilities diverge. The details of conditions for cloning states in the model can be referred to (Borges & Levene 2004). Basically, if there is considerable difference between first and second-order probabilities of a state, its in-links will be divided into clusters based on the second-order probabilities, and state clones are created with respect to the in-link clusters.

6.2.4. Reasoning Algorithms for the Web-page Navigation Model

This sub-section will present two reasoning algorithms based on the Web-page navigation model. These algorithms will be used to predict Web-pages in the recommendation process in the Web-page recommender system of this research.

Algorithm 6-2: Query about next pages for a given current page and a given previous page

In the context of Web-page recommendation, we might need to query next visited pages for a given currently visited page *curP* and a given previously visited page *preP*. We can easily achieve this based on a second-order WPNavNet O_D by applying an algorithm, named ***RecPage*(*preP, curP, O*)**, as shown in Algorithm 6-2.

This algorithm is developed basing on the second-order Web-page navigation model. It firstly finds a *cNode* instance which satisfies the following conditions: (i) named *curP*, and (ii) has an *inLink* being *preP*, then retrieves *cOutLink* instances that are associated with the *cNode* instance via the *outLink* object property. The names of the found *cOutLink* instances are the predicted next pages. In order to obtain most frequent pages for later Web-page recommendation, the list of predicted pages is sorted in descending order of transition probabilities assigned in the *cOutLink* instances. Based on Definitions 6.1-6.5, the algorithm is described in logics notation as follows:

$RecPage(d_x, d_y) = \{d_1, d_2, \ldots, d_s\}$,
where
$d_y \in D$: the current page,
$d_x \in D$: the previous page,
(d_x, d_y, d_i) is a transition in $\mathcal{M}_D$, $d_i \in D$, $1 \le i \le s$, and
$\varphi_{x,y,i} > \varphi_{x,y,j}$, $(i < j$ and $1 \le i, j \le s)$.

Algorithm 6-2: Query about next pages for a given current page and a given previous page

Input:
preP (the previous page)
curP (the current page)
O_D (2^{nd}-order WPNavNet)
Output: recP (recommended pages)
Process:
Traverse through *O* to get the *cNode* instances whose name is *curP*
For each instance *curP*
Set *inS* = the *cNode* instances linked to instance *curP* as inlinks
If *inS* contains *preP* then *recP* = *cOutLink* instances associated with *curP* via the *outLink* object property
Sort *recP* in descending order of the probabilites of *cOutLink* instances
Return *recP*

Algorithm 6-3: Query about next pages for a given current page

To recommend next pages for a given page currently visited by a user, the first-order

Web-page navigation model is used. In this case, an algorithm, as Algorithm 6-3, is applied to retrieve next visited pages given a currently visited page *curP* based on a first-order WPNavNet O_D, named ***RecPage*(*curP, O*)**. This algorithm is designed in a similar way as Algorithm 6-2, except that the first-order transition probabilities are applied and the previously visited page *preP* and the *inLink* object properties are not taken into account. Based on Definitions 6.1-6.5, the algorithm is described in logics notation as follows:

$RecPage(d_x) = \{d_1, d_2, \ldots, d_s\}$,
where
$d_x \in D$: the current page,
(d_x, d_i) is a transition in Φ_D, $d_i \in D$, $1 \le i \le s$, and
$\varphi_{x,i} > \varphi_{x,j}$, $(i < j$ and $1 \le i, j \le s)$.

Algorithm 6-3: Query about next pages for a given current page

Input:
curP (the current page)
O_D (1st-order WPNavNet)
Output: recP (recommended pages)
Process:
Traverse through *O* to get the *cNode* instance whose name is *curP*
Set *recP* = *cOutLink* instances associated with *curP* via the *outLink* object property
Sort *recP* in descending order of the probabilites of *cOutLink* instances
Return *recP*

These reasoning algorithms merely predict Web-pages without considering the domain terms of Web-page. To semantically enhance Web-page recommendation, the next section presents a new navigation model involving the domain terms.

6.3. A Semantic Web Usage Knowledge Representation Model for Web-Page Recommendation

Chapter 4 has presented a model of knowledge representation, DomainOntoWP, to capture the domain knowledge of a website for supporting Web-page recommendation, while Chapter 5 has presented another model of knowledge representation, TermNetWP, to capture the semantics of Web-pages within a website. Although they are efficient for capturing the domain knowledge and semantics of a given website, they are not sufficient

on their own for making effective Web-page recommendations. Besides, the Web usage knowledge, namely FWAP, can be used for Web-page recommendation. However, the system merely recommends next pages by matching Web-page patterns in FWAP, but cannot understand what they are about. In order to make better Web-page recommendations, it is essential to take into account semantic Web usage knowledge so that what topics are viewed on Web-pages can be understood. To obtain this semantic knowledge, the domain knowledge model (DomainOntoWP) or the semantic network (TermNetWP) needs to be integrated with the FWAP. The integration of the FWAP with DomainOntoWP or TermNetWP results in a set of frequently viewed domain term patterns (**FVTP**), as shown in Figure 6-3.

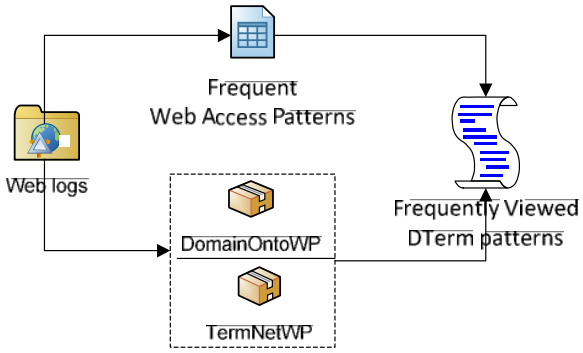


Figure 6-3: Frequently viewed domain term pattern discovery

Description of the FVTP is formalized in the following definitions.

Definition 6.6 Let $D = \{d_j : 1 \le j \le q\}$ be a set of Web-pages, $\mathcal{T} = \{t_i : 1 \le i \le p\}$ be a set of domain terms in the titles of the Web-pages, where $t_i \in T_{man}$ if DomainOntoWP is involved, or $t_i \in T_{auto}$ if TermNetWP is involved. Let $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ be a set of FWAP, where each pattern P_i (i = [1..n]) contains a sequence of Web-pages, n is the number of the patterns, and $P_i = d_{i1}d_{i2} \ldots d_{im}$, $d_{ik} \in D$, k = [1..m], m is the number of Web-pages in the pattern.

Definition 6.7 Based on Definition 6.6, since each page $(d_{ik}) \subseteq P_i$ ($P_i \in \mathcal{P}$) contains a set of

domain terms $\tau \subset \mathcal{T}$, we can generate a set of **FVTP** $\mathcal{F} = \{t_{i1}t_{i2} \ldots t_{im} : t_{ik} \in \mathcal{T} \wedge i = [1..n] \wedge k = [1..m]\}$, where each domain term pattern $F = t_{i1}t_{i2} \ldots t_{im}$ is a sequence of domain terms, in which each domain term t_{ik} is a domain term of page d_{ik} in P_i.

In order to obtain the semantic Web usage knowledge that is efficient for semantic-enhanced Web-page recommendation, FVTP is represented in an effective domain term navigation model, which will be presented in the following sub-sections.

6.3.1. A Domain Term Navigation Model

A domain term navigation model is proposed to automatically generate a weighted semantic network of frequently viewed domain terms with the weight being the probability of the transition between two adjacent domain terms based on FVTP. This weighted semantic network is the semantic Web usage knowledge of a website for Web-page recommendation. We refer to this domain term navigation network as TermNavNet hereafter. Figure 6-4 illustrates how the domain term navigation model acts as a formatter to convert FVTP into TermNavNet.

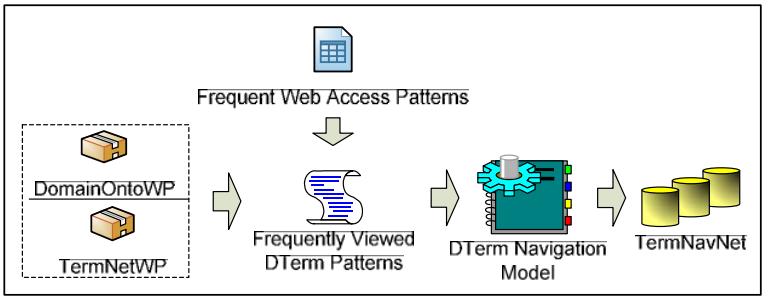


Figure 6-4: Building a domain term navigation network

Similar to the Web-page navigation model, the domain term navigation model has two main kinds of elements: (1) state nodes; (2) the relations between state nodes. One state node presents the current state, e.g. the domain term currently viewed by a user (or the current domain term for short), and may have some previous state nodes, e.g. domain terms previoustly viewed by a user (or previous domain terms for short), and some next state nodes, e.g. next domain terms viewed by a user (or next domain terms for short). By

scanning each domain term pattern $F \in \mathbb{F}$, each domain term becomes a state node in the model. There are also two additional states: a start state, *S*, representing the first state of every domain term pattern; and a final state, *E*, representing the last state of every domain term pattern. There is a transition corresponding to each pair of domain terms in a pattern, a transition from the start state *S* to the first domain term of a domain term pattern, and a transition from the last domain term of a domain term pattern to the final state *E*. The model is incrementally built by processing the complete collection of domain term patterns.

The formal definitions that are used to formally describe the domain term navigation model are listed as follows:

Definition 6.8 Given the domain term pattern set $\mathbb{F}$ and a domain term $t_x \in \mathbb{T}$, we define $\partial_x = |\{F : F \in \mathbb{F} \wedge (t_x) \subseteq F\}|$ as the number of occurrences of t_x in $\mathbb{F}$. Given the domain term pattern set $\mathbb{F}$ and a pair of domain terms (t_x, t_y), $t_x, t_y \in \mathbb{T}$, we define $\partial_{x,y} = |\{F : F \in \mathbb{F} \wedge (t_x t_y) \subseteq F\}|$ as the number of times that t_x is followed by t_y in $\mathbb{F}$ and there is no term between them. We also define $\partial_{S,x}$ as the number of times that domain term $t_x \in \mathbb{T}$ is the first item in a domain term pattern $F \in \mathbb{F}$, and $\partial_{x,E}$ as the number of times that a domain term pattern $F \in \mathbb{F}$ terminates at domain term $t_x \in \mathbb{T}$. Furthermore, given the domain term pattern set $\mathbb{F}$ and a transition (t_x, t_y, t_z), $t_x, t_y, t_z \in \mathbb{T}$, we define $\partial_{x,y,z} = |\{F : F \in \mathbb{F} \wedge (t_x t_y t_z) \subseteq F\}|$ as the number of times that (t_x, t_y) is followed by t_z in $\mathbb{F}$ and there is no term between them.

Based on Definition 6.8, the first and second-order transition probabilities are estimated in a similar way as Definitions 6.3 and 6.4, respectively.

Definition 6.9 **(Domain term navigation model)** By Definitions 6.6, 6.7, and 6.8, and the definitions of the first and second-order transition probabilities (6.3 and 6.4), the domain term navigation model is formalized as a triple:

$$O_T := <\mathbb{N}_T, \boldsymbol{\Phi}_T, \mathbb{M}_T>, \tag{6.6}$$

where

$\mathbb{N}_T = \{(t_x, \partial_x) : t_x \in \mathbb{T}\}$: a set of domain terms along with the corresponding occurrence

counts,

$\Phi_T = \{(t_x, t_y, \partial_{x,y}, \rho_{x,y}) : t_x, t_y \in T\}$: a set of transitions from t_x to t_y, along with their transition weights ($\partial_{x,y}$), and first-order transition probabilities ($\rho_{x,y}$),

$M_T = \{(t_x, t_y, t_z, \partial_{x,y,z}, \rho_{x,y,z}) : t_x, t_y, t_z \in T\}$: a set of transitions from t_x, t_y to t_z, along with their transition weights ($\partial_{x,y,z}$), and second-order transition probabilities ($\rho_{x,y,z}$).

If M_T is non-empty, the model (6.6) is considered as *the second-order domain term navigation model*, otherwise *the first-order domain term navigation model*.

6.3.2. Schema of Domain Term Navigation Model

In order to automatically construct TermNavNet for a given FVTP, this sub-section designs the schema of the domain term navigation model in the similary way as the schema of the Web-page navigation model, and then implements it in OWL. Its schema consists of classes *cNode* and *cOutLink*, and the relationships between them, namely *inLink*, *outLink*, and *linkTo*, as shown in Figure 6-5, where *cNode* and *cOutLink* define the current state node and the association with a next state node, respectively. Each state node represents a domain term in FVTP.

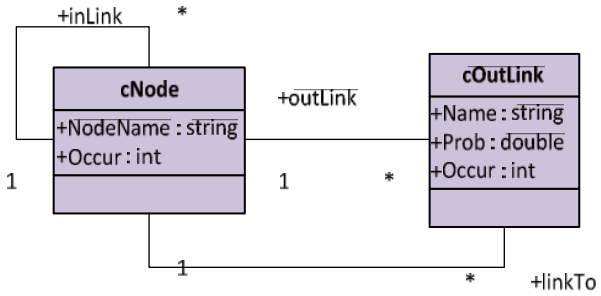


Figure 6-5: The schema of domain term navigation model

Class *cNode* represents a domain term in the domain term navigation model, and that domain term is assigned to its *NodeName* property value. Class *cNode* has two *inLink* and *outLink* object properties referring to *cNode* and *cOutLink*, respectively. The number of occurrences of the domain term is represented by *Occur*, i.e. ∂_x. *inLink* represents an

association from a previous state node, e.g. a previously viewed domain term, to the state node it belongs to. Class *cOutLink* represents an association from a state node to another next state node with a transition probability *Prob*, e.g. $\rho_{x,y}$, and the number of the associations is captured by its *Occur* property. Class *cOutLink* has a *linkTo* object property referring to a *cNode* object, i.e. the next state node, so the name of this *cNode* object is assigned to the *Name* property of *cOutLink*.

6.3.3. Automatic Construction of Domain Term Navigation Model

The domain term navigation model for a website, i.e. TermNavNet, can be automatically constructed by populating the schema of the domain term navigation model with the given set of FVTP. An algorithm is designed to accomplish this task, as shown in Algorithm 6-4.

Algorithm 6-4: TermNavNet construction

Input: $\mathbb{F}$(FVTP)
Output: *M* (TermNavNet)
Process:
  Initialize *M*
  For each $F = t_1...t_m \in \mathbb{F}$
    For each $t_i \in F$, $i = [1..m]$
      Initialize *cNode* objects with *NodeName* = t_i, t_{i-1}, t_{i+1} and *Occur* = *1* if they are not found in *M*
      Initialize a *cOutLink* object with *Name* = $t_i\_t_{i+1}$ and *Occur* = *1* if it is not found in *M*
      Increase t_i.*Occur* and $t_i\_t_{i+1}$.*Occur* if they are found in *M*
      $t_i\_t_{i+1}$.*linkTo* = t_{i+1}
      t_i.*outLink* = $t_i\_t_{i+1}$
      t_i.*inLink* = t_{i-1}
      Update all objects into *M*
  Update transition probabilities in the *cOutLink* objects
Return *M*

The transition probabilities in the *cOutLink* objects can be updated based on the first-order or second-order probability formulae, i.e. (6.1-6.3) or (6.4), depending on which order of the domain term navigation model is applied. Similar to the Web-page navigation model, the second-order domain term navigation model is built by using the dynamic clustering-based higher-order Markov model (Borges & Levene 2004) to elliviate the complexity of the second-order model. Consequently, a first-order or second-order TermNavNet are built.

6.3.4. Reasoning Algorithms for the Domain Term Navigation Model

Similar to the reasoning algorithms of Web-page navigation model presented in Section 6.2.4, the reasoning algorithms of domain term navigation model are applied to the domain terms of Web-pages instead of Web-pages. This sub-section will present four reasoning algorithms based on the domain term navigation model.

Algorithm 6-5: Query about next domain terms for a given current domain term and a given previous domain term

For Web-page recommendation, we might need to find out the next viewed domain terms for a given currently viewed domain term *curT* and a given previously viewed domain term *preT*. We can easily achieve this based on a second-order TermNavNet O_T by applying an algorithm, named ***RecDTerm(preT, curT, O_T)***, as shown in Algorithm 6-5. This algorithm is designed basing on the second-order domain term navigation model in the similar way as Algorithm 6-2. Based on Definition 6.9, the algorithm is described in logics notation as follows:

$RecDTerm(t_x, t_y) = \{t_1, t_2, \ldots, t_s\}$,
where
$t_y \in \mathcal{T}$: the current domain term,
$t_x \in \mathcal{T}$: the previous domain term,
(t_x, t_y, t_i) is a transition in $\mathcal{M}_T$, $t_i \in \mathcal{T}$, $1 \le i \le s$, and
$\rho_{x,y,i} > \rho_{x,y,j}$, ($i < j$ and $1 \le i, j \le s$).

Algorithm 6-5: Query about next domain terms for a given current domain term and a given previous domain term

Input:
preT (the previous domain term)
curT (the current domain term)
O_T (2^{nd}-order TermNavNet)
Output: recT (recommended domain terms)
Process:
Traverse through O_T to get the *cNode* instances whose name is *curT*
For each instance *curT*
Set *inS* = the *cNode* instances linked to instance *curT* as inlinks
If *inS* contains *preT* then *recT* = *cOutLink* instances associated with *curT* via the *outLink* object property
Sort *recT* in descending order of the probabilites of *cOutLink* instances
Return *recT*

Algorithm 6-6: Query about next domain terms for a given current domain term

To recommend the next domain terms for a given page currently visited by a user, the first-order domain term navigation model is used. To do this task, an algorithm, as Algorithm 6-6, is designed in a similar way as Algorithm 6-3. Specifically, it retrieves the next viewed domain terms given each currently viewed domain term *curT* of the current page based on a first-order TermNavNet O_T, named ***RecDTerm*(*curT*, O_T)**. Based on Definition 6.9, the algorithm is described in logics notation as follows:

$RecDTerm(t_x) = \{t_1, t_2, \ldots, t_s\}$,
where
$t_x \in \mathcal{T}$: the current domain term,
(t_x, t_i) is a transition in Φ_T, $t_i \in \mathcal{T}$, $1 \le i \le s$, and
$\rho_{x,i} > \rho_{x,j}$, ($i < j$ and $1 \le i, j \le s$).

Algorithm 6-6: Query about next domain terms for a given current domain term

Input:
curT (the current domain term)
O_T (1st-order TermNavNet)
Output: recT (recommended domain terms)
Process:
Traverse through O_T to get the *cNode* instance whose name is *curT*
Set *recT* = *cOutLink* instances associated with *curT* via the *outLink* object property
Sort *recT* in descending order of the probabilites of *cOutLink* instances
Return *recT*

Algorithm 6-7: Query about next domain terms and respective prediction probabilities for a given current domain term and a given previous domain term

It may be necessary to obtain the next viewed domain terms with respective prediction probabilities given a currently viewed domain term and a previously viewed domain term for making more effective Web-page recommendations. An algorithm, as Algorithm 6-7, is designed in the similar way as Algorimth 6-5, which takes the given current domain term, i.e. *curT*, and a previous domain term, i.e. *preT*, as the inputs to query for the next domain terms and their respective prediction probabilities based on a second-order TermNavNet O_T, named ***RecDTermProb*(*preT*, *curT*, O_T)**. This algorithm retrieves *cOutLink* instances that are associated with the *cNode* instance which satisfies the following conditions: (i)

named *curT*, and (ii) has an *inLink* being *preT*. The returned results include the next viewed domain terms, i.e. *cOutLink* instance names, along with respective prediction probabilities, i.e. the *Prob* values of respective *cOutLink* instances. Based on Definition 6.9, the algorithm is described in logics notation as $RecDTermProb(t_x, t_y) = \{(t_z,\rho) : (t_x, t_y, t_z, \partial, \rho) \in \mathcal{M}_T \wedge t_x, t_y, t_z \in \mathcal{T}\}$, where t_x is the previous domain term, and t_y is the current domain term.

Algorithm 6-7: Query about next domain terms and respective prediction probabilities for a given current domain term and a given previous domain term

Input:
  preT (the previous domain term)
  curT (the current domain term)
  O_T (2nd-order TermNavNet)
Output: *recT* (collection of predicted next domain terms & respective probabilities)
Process:
  Traverse through O_T to get the *cNode* instances whose name is *curT*
  For each instance *curT*
    Set *inS* = the *cNode* instances linked to instance *curT* via the *inLink* object property
    If *inS* contains *preT* then
      Set *outS* = *cOutLink* instances associated with the *cNode* instance via the *outLink* object property
      Set *recT* = the names of *outS* with the respective prediction probabilities
    Return *recT*

Algorithm 6-8: Query about next domain terms and respective prediction probabilities for a given current domain term

This algorithm, as Algorithm 6-8, queries for next viewed domain terms with respective prediction probabilities given a currently viewed domain term *curT* based on a first-order TermNavNet O_T, named ***RecDTermProb(curT, O_T)***. Based on Definition 6.9, the algorithm is described in logics notation as $RecDTermProb(t_x) = \{(t_y,\rho) : (t_x, t_y, \partial, \rho) \in \Phi_T \wedge t_x, t_y \in \mathcal{T}\}$, where t_x is the current domain term.

Algorithm 6-8: Query about next domain terms and respective prediction probabilities for a given current domain term

Input:
  curT (the current domain term)
  O_T (1st-order TermNavNet)
Output: *recT* (collection of predicted next domain terms & respective probabilities)
Process:
  Traverse through O_T to get the *cNode* instance whose name is *curT*
  Set *outS* = *cOutLink* instances associated with the *cNode* instance via the *outLink* object property
  Set *recT* = the names of *outS* with the respective prediction probabilities
  Return *recT*

The reasoning algorithms of the domain term navigation model enable to predict Web-pages based on domain terms because there is the mapping between domain terms and Web-pages in the domain knowledge representation models, i.e. DomainOntoWP, and TermNetWP. This prediction task will be explained by an example in the following section.

6.4. An Example of Using the Proposed Navigation Models

This section illustrates an example of applying the Web-page navigation model and the domain term navigation model to a sample set of FWAP, and shows how the reasoning algorithms of the models can be used for Web-page prediction. The sample set of FWAP, which is discovered from the Microsoft (MS) Web data using PLWAP-Mine, is shown in Figure 6-6, where each number refers to a Web-page ID. The MS Web data is available at http://kdd.ics.uci.edu/databases/msweb/msweb.html.

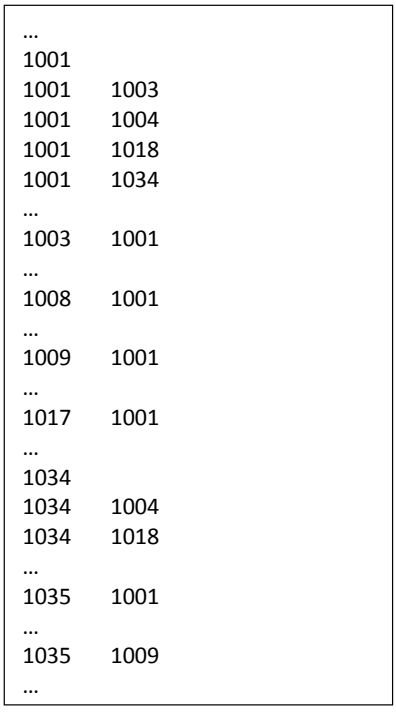


Figure 6-6: A sample set of frequent Web access patterns

Firstly, Web-page navigation networks are built for the given sample FWAP using the Web-page navigation model. There are two kinds of Web-page navigation networks, i.e. the first-order WPNavNet and the second-order WPNavNet, which are generated with respect to the first-order and second-order Web-page navigation models. Figure 6-7 depicts a part of the first-order WPNavNet which contains the state for page 1001 and its related states. Its previous states represent the pages that were visited before this page, such as 1003, 1008, 1009, 1017, 1035. Its next states represent the pages that may be visited in the next step from this page, such as 1003, 1004, 1018, 1034. The numbers on the out-links of page 1001 are the estimated first-order transition probabilites. Since pages 1004 and 1018 are visited after page 1034, as shown in Figure 6-6, there are transitions from state 1034 to states 1004 and 1018 with the corresponding first-order transition probabilites shown.

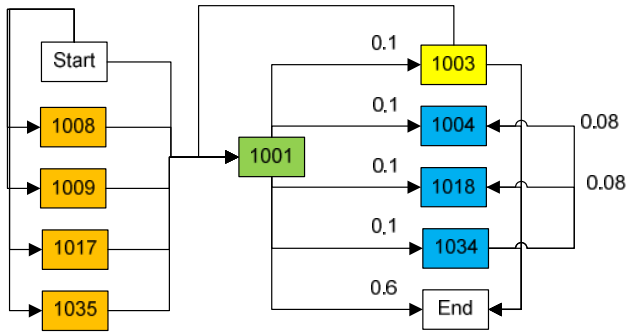


Figure 6-7: A sample 1st-order WPNavNet

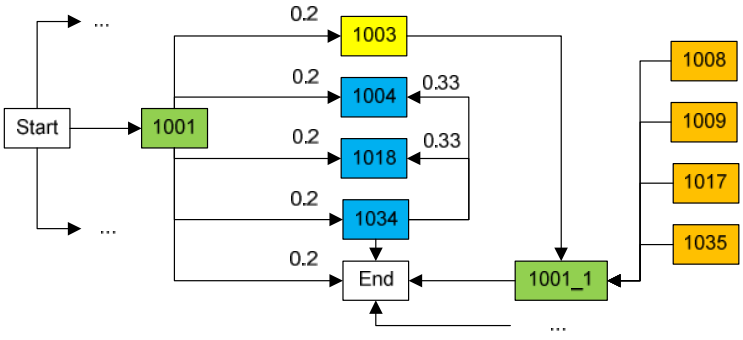


Figure 6-8: A sample 2nd-order WPNavNet

Figure 6-8 depicts a part of the second-order WPNavNet which contains the state for page 1001 and its related states. For this second-order WPNavNet, the state 1001 is cloned because its first and second-order probabilities diverge. The second-order transition probability from *Start* to 1001, then to 1003, 1004, 1018, or 1034 is equal to 0.2, while the first-order transition probability from 1001 to 1003, 1004, 1018, or 1034 is 0.1. The second-order transition probability from 1003, 1008, 1009, 1017, or 1035 to 1001, then to *End* is equal to 1, while the first-order transition probability from 1001 to *End* is 0.6. Hence, the in-links of 1001 are divided into two clusters: (1) {Start}, and (2) {1003, 1008, 1009, 1017, 1035}. The first in-link cluster is connected to 1001, and followed by 1003, 1004, 1018, and 1034. The second in-link cluster is connected to a 1001's clone, named 1001_1, and followed by *End*. The transition probabilities of 1001 and its clone are then computed again, as shown in Figure 6-8.

Each state in the first-order navigation network and the second-order navigation network has the same set of in-links and the same set of out-links, but depending on which previous state is traversed, the next state will be predicted differently in the two navigation networks.

Based on these networks, we can predict next pages for a given previous page and a given current page using the reasoning algorithms in Section 6.2.4.

Secondly, domain term navigation networks for the MS website are built using the domain term navigation model. There are two kinds of domain term navigation networks, i.e. the first-order TermNavNet and the second-order TermNavNet, which are generated with respect to the first-order and second-order domain term navigation models. To do this task, we need to generate FVTP from the FWAP integrated with the domain knowledge which is represented based on the titles of Web-pages, as described in Section 6.3. Figure 6-9 shows a sample set of Web-pages and titles in the MS website. In this example, the TermNetWP model is integrated with the FWAP to generate FVTP.

PageID	Title
1001	Support Desktop
1003	Knowledge Base
1004	Microsoft.com Search
1008	Free Downloads
1009	Windows Family of OSs
1017	Products
1018	isapi
1026	*Internet Site Construction for Developers*
1034	Internet Explorer
1035	Windows95 Support
…	

Figure 6-9: A sample set of Web-pages and titles

Given the generated FVTP, the TermNavNets are constructed. For example, a sample second-order TermNavNet is shown in Figure 6-10. Since pages 1003, 1004, 1018, and 1034 are visited after page 1001, their domain terms *Knowledge_base*, *Microsoft.com_Search*, *isapi*, *Internet*, and *Explorer* are the out-links of domain terms *Support* and *Desktop* in the page 1001 title. The numbers on the out-links describe the prediction probabilities. Furthermore, domain term *Support* is followed by the domain terms *Windows* and *Family_OSs* of page 1009 as well, because page 1035, including term *Support*, is followed by page 1009 in the sample FWAP.

This graphical sample (Figure 6-10) is similar for the first-order TermNavNet, but the prediction probabilities of the first-order TermNavNet can be different from those of the second-order TermNavNet.

Based on this network, we can predict Web-pages using the reasoning algorithms in Section 6.3.4 with the help of TermNetWP. That is TermNetWP is used to query for the domain terms of the visited Web-pages, TermNavNet is used to predict next domain terms from these domain terms, then TermNetWP is used to query for the Web-pages mapped to the predicted domain terms.

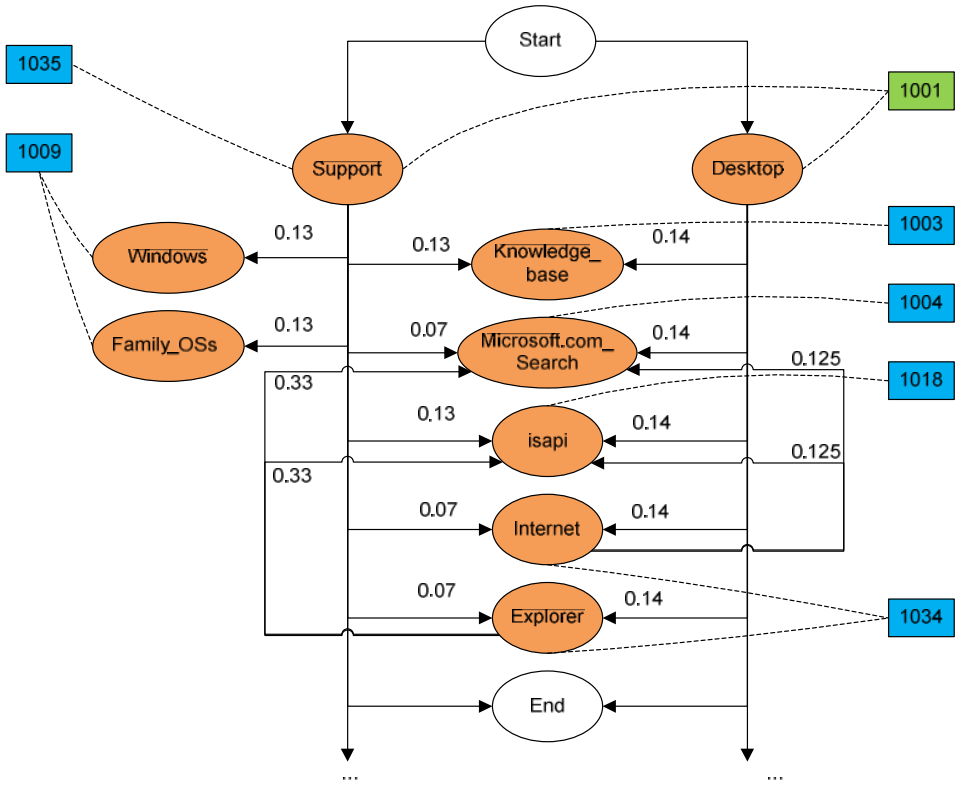


Figure 6-10: A sample TermNavNet

Finally, some Web-page prediction cases are presented using the above navigation networks in Table 6-2.

Table 6-2: Web-page prediction cases

Previous page	Current page	Predicted next pages			
		1st WPNavNet	2nd WPNavNet	1st TermNavNet	2nd TermNavNet
1008	1001	1003, 1004, 1018, 1034	-	1003, 1004, 1018, 1034, *1009*	-
-	1001	1003, 1004, 1018, 1034	1003, 1004, 1018, 1034	1003, 1004, 1018, 1034, *1009*	1003, 1004, 1018, 1034, *1009*
-	1026	-	-	1004, 1018	1004, 1018

Based on Table 6-2, the following facts can be observed:

- If a user visits page 1001 after visiting page 1008, pages 1003, 1004, 1018, and 1034

are recommended based on the first-order WPNavNet, while no page is recommended based on the second-order WPNavNet. On the other hand, the first-order TermNavNet can recommend the same pages and page 1009. This is because the domain term *Support* of page 1001 is linked to the domain terms of page 1009 as its out-links (Figure 6-10). It shows that the TermNavNet model can enrich the pool of recommended Web-pages.

- If a user visits page 1001 first, that is the previous state is *Start*, both first-order and second-order WPNavNet provide the same recommended pages. That means with the same current state, 1001, but the previous state is different from the previous observation, the recommended results will be different for the second-order WPNavNet. This also shows that the second-order model is more dependent on the previous state than the first-order model. Therefore, the predicted page set by the second-order model is always a subset of the predicted page set by the first-order model.

- If a user visits page 1026 first, both first-order and second-order WPNavNet cannot recommend any pages because there is not this page in both networks. This is a limitation of the WPNavNet model. In contrast, the TermNavNets can offer the predicted next pages, e.g. 1004, and 1018. Because page 1026 includes domain term *Internet* (Figure 6-9), domain terms *Microsoft.com_Search*, and *isapi* are predicted based on domain term *Internet* (Figure 6-10). This is the benefit of the TermNavNet model, since it enables to predict Web-pages for a given new page which is about the domain terms of interest.

6.5. Summary

This chapter has presented the two new concept navigation models based on the Markov models to represent the Web usage knowledge of a given website for Web-page recommendation. The models are implemented in an ontological fashion using OWL to facilitate knowledge sharing, reuse and integration. They are useful to predict the next

pages the users might visit for improving Web-page recommendation.

Chapter 7.

A SEMANTIC-ENHANCED WEB-PAGE RECOMMENDER SYSTEM FRAMEWORK

7.1. Introduction

Based on the two models of representing domain knowledge, i.e., DomainOntoWP proposed in Chapter 4, and TermNetWP proposed in Chapter 5, and the two models of representing Web usage knowledge, i.e., WPNavNet and TermNavNet proposed in Chapter 6, this chapter proposes a detailed framework of the semantic-enhanced Web-page recommender system (SWRS) coordinating the two knowledge types of domain and Web usage. Based on the reasoning algorithms developed on these knowledge representation models, a set of Web-page recommendation strategies are thoughtfully developed to make semantic-enhanced Web-page recommendations. To validate the knowledge representation models, i.e. DomainOntoWP, TermNetWP, WPNavNet and TermNavNet, and to evaluate the reasoning algorithms, as well as the recommendation strategies, a number of experiments of Web-page recommendation are carried out using the proposed models, compared with a Web-page recommendation model using an advanced existing Web usage mining method, which is Pre-Order Linked WAP-Tree Mining (PLWAP-Mine for short) (Ezeife & Lu 2005).

This chapter is structured as follows: Section 7.2 presents a detailed framework for the semantic-enhanced Web-page recommender system of this research. Section 7.3 proposes a set of recommendation strategies based on the reasoning algorithms of the proposed knowledge representation models to make effective Web-page recommendations. Section 7.4 presents a set of detailed experiments to evaluate the performance of the proposed models, algorithms and strategies, along with the analysis of experimental results. Some discussions are given in Section 7.5, and a summary of this chapter is provided in Section 7.6.

7.2. Framework

This section proposes a framework of the semantic-enhanced Web-page recommender system which integrates the domain knowledge representation models, i.e. DomainOntoWP, or TermNetWP, with the Web usage knowledge representation models, i.e. WPNavNet, and TermNavNet, for Web-page recommendation. Figure 7-1 illustrates the workflow of this framework, which consists of five stages: Stages 1-4 are conducted off-line, and Stage 5 is carried out on-line. The five stages are processed respectively by five components, which are Pre-processing unit, Web usage mining (WUM) unit, Domain knowledge construction unit, Prediction model unit, and Web-page recommendation generation unit. The input data of the system is a Web log of a website. In off-line stages 1-4, the data is processed and modeled to have useful knowledge bases for making Web-page recommendation. In on-line stage 5, the modeled knowledge is used to query Web-pages which are recommended to an active user given the user's current Web access. Stages 1-5 are stated in the following details.

Stage 1: Pre-processing

This is a data preparation stage, that is, data is collected, integrated from multiple sources, and transformed into forms suitable for input into the next processes, i.e. Web usage mining and domain knowledge construction. The main data used in the SWRS is Web usage data which is hidden in Web server logs implicitly recording the browsing history of users. Web log files are able to be collected from the Web server of a given website. The Web log file needs to be cleaned in order to extract useful information: (1) datasets of users' Web usage sessions, and (2) a list of accessed Web-page paths (URLs). Given the datasets of Web usage, we can acquire Web access sequences (WAS). A WAS is a sequence of items (or events) which are the identification numbers of Web-pages visited by a user in a session. The all URLs are crawled to retrieve the metadata of Web-pages, namely the accessed Web-page titles which are concerned in this research, and are clues to describe the semantics of Web-pages.

Stage 2: Web Usage Mining

In this stage, PLWAP-Mine (Ezeife & Lu 2005) is used to analyse and discover Web usage patterns. The process is performed in the same way as described in the sequential pattern mining component in the Web-page recommender system architecture in Chapter 3. As a result, a complete set of frequent Web access patterns (FWAP) is discovered as useful Web usage knowledge for Web-page prediction in the prediction model and Web-page recommendation generation stages.

Stage 3: Domain Knowledge Construction

There are two models of domain knowledge construction of the given website. The domain ontology model presented in Chapter 3 is used to construct the domain ontology of terms, Web-pages and the relationships between them, which is DomainOntoWP. The semantic network model presented in Chapter 4 is used to automatically construct a semantic network of Web-pages mapping to domain terms, which is TermNetWP. One of the DomainOntoWP or TermNetWP models is used for supporting effectively Web-page recommendation later on.

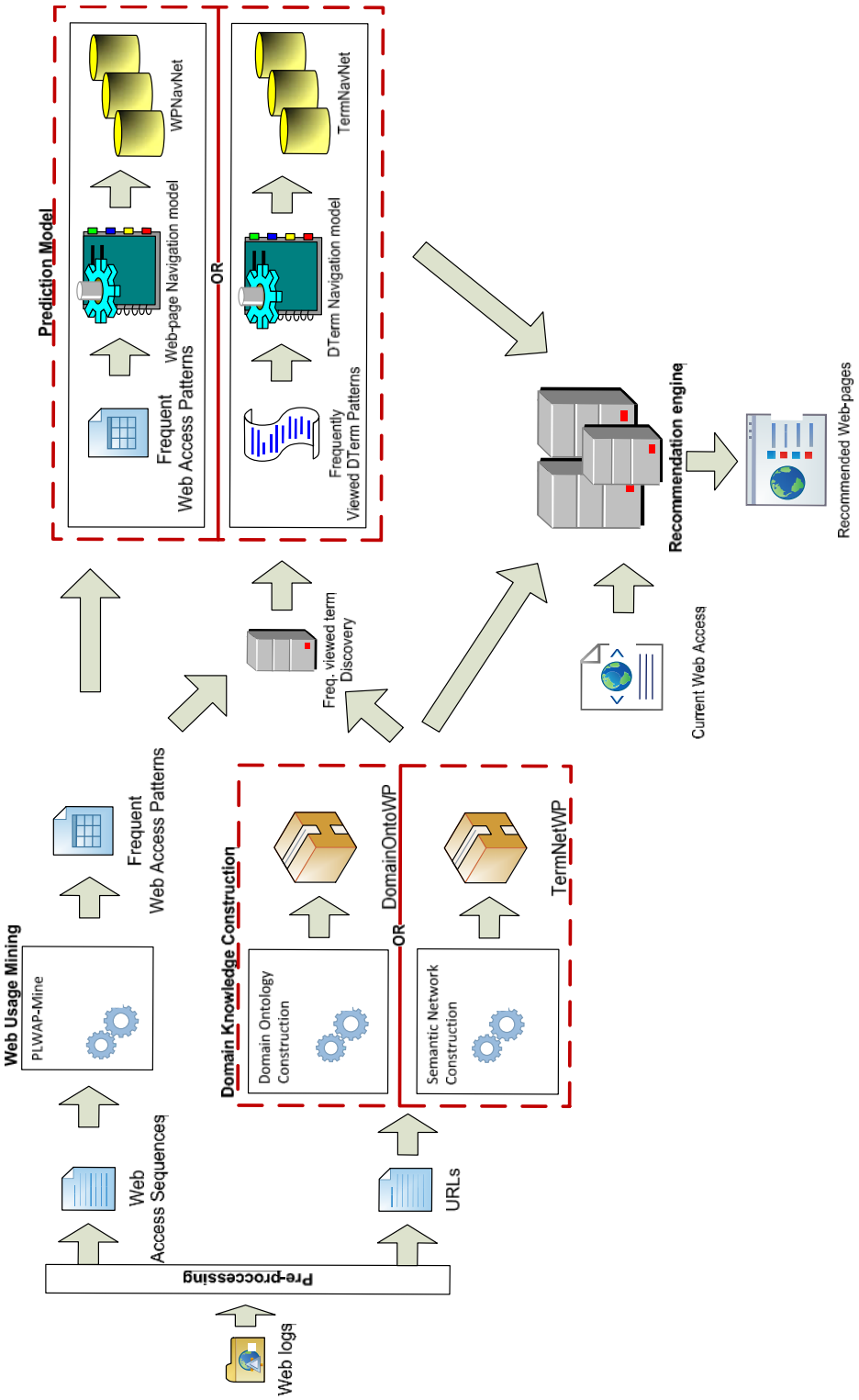


Figure 7-1: Framework of the semantic-enhanced Web-page recommender system

Stage 4: Prediction Model

There are two kinds of concept navigation models built for the prediction process, which are proposed in Chapter 6. The first model, which is the Web-page navigation model, adopts FWAP to automatically generate WPNavNet. The second model, which is the domain term navigation model, adopts FWAP integrated with the domain knowledge, to automatically generate TermNavNet. The integrated domain knowledge is DomainOntoWP or TermNetWP. This integration results in a set of frequently viewed domain term patterns to be transformed into TermNavNet, namely the DomainOntoWP-based TermNavNet or the TermNetWP-based TermNavNet. Both models, i.e. WPNavNet and TermNavNet, are able to predict Web-pages and domain terms, respectively, and they need to be combined with the domain knowledge representation model to make semantic-enhanced Web-page recommendations.

Stage 5: Web-page Recommendation Generation

Web-page recommendation generation is performed based on the combination of the transformed Web usage knowledge, i.e. WPNavNet and TermNavNet, and the modeled domain knowledge, i.e. DomainOntoWP and TermNetWP. To accomplish the task, three types of recommendation strategies are proposed based on the developed reasoning algorithms of the knowledge representation models. The first strategy type, which is the Web-page recommendation without semantic enhancement, uses WPNavNet to make Web-page recommendations. The second strategy type, which is the Web-page recommendation with semantic enhancement based on the domain ontology, coordinates WPNavNet, and the DomainOntoWP-based TermNavNet to make Web-page recommendations. The third strategy type, which is the Web-page recommendation with semantic enhancement based on the semantic network, coordinates WPNavNet, and the TermNetWP-based TermNavNet to make Web-page recommendations.

Based on one of these strategies, given a current Web access sequence generated when a user browses the website, the recommendation engine can generate next Web-pages to be recommended to the user. Table 7-1 briefly describes the procedure of Web-page recommendation generation for a given user's Web access sequence.

Table 7-1: Procedure of Web-page recommendation generation

<table>
<tr><td></td><td colspan="2">(1) Input a user's Web access sequence, the last one/two visited Web-pages in the Web access sequence are taken into account;</td></tr>
<tr><td>Without semantic enhancement</td><td colspan="2">(2) Traverse WPNavNet in the Prediction model unit to find Web-pages matching with the last visited Web-pages;
- If found, query for the next Web-pages;
- If not found, or the number of next Web-pages is less than a required number, do the next steps;</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td rowspan="4">With semantic enhancement</td><td>DomainOntoWP-based</td><td>TermNetWP-based</td></tr>
<tr><td>(3) Traverse DomainOntoWP in the Domain knowledge construction unit to find Web-page instances matching with the last visited Web-pages, and to query for the domain terms of these Web-pages;</td><td>(3) Traverse TermNetWP in the Domain knowledge construction unit to find Web-page instances matching with the last visited Web-pages, and to query for the domain terms of these Web-pages;</td></tr>
<tr><td colspan="2">(4) If only the last visited page is taken into account,
- For each term of this page, traverse TermNavNet in the Prediction model unit to find a domain term instance matching with this term, and to query for the next domain terms;
If the last two visited pages, which are the previously visited page and the currently visited page, are taken into account,
- For each combination of one term of the previously visited page and one term of the currently visited page, traverse TermNavNet to find term instances matching with these two terms, and to query for the next domain terms;</td></tr>
<tr><td>(5) For each next term, traverse DomainOntoWP to find a term instance matching with this term, and to query for pages mapping to this term instance;</td><td>- For each next term, traverse TermNetWP to find a term instance matching with this term, and to query for pages mapping to this term instance;
- Furthermore, it is possible to combine the next terms corresponding to each term or each combination of the two terms with respect to the last two visited pages, or to combine all the next terms to query for pages mapping to these terms in TermNetWP;</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td></td><td colspan="2">(6) Recommend the Web-pages queried in Steps (2) and (5) to the user.</td></tr>
</table>

Table 7-1 shows how the domain and Web usage knowledge bases are coordinated with each other in the system. The power of SWRS is presented through the semantic-enhanced recommendation strategies, in which WPNavNet is applied firstly in order to produce early recommended next Web-pages, and then TermNavNet is applied to offer more recommended next Web-pages. TermNavNet will play an important role in case WPNavNet cannot make recommendation as the last visited pages do not exist in the WPNavNet. Details about the recommendation strategies will be presented in Section 7.3.

7.3. Web-page Recommendation Strategies

This section proposes various Web-page recommendation strategies based on the proposed domain and Web usage knowledge representation models. For a given current Web-page or the given combination of the current and previous Web-pages taken by a user, next Web-pages are recommended to the user based on the modeled knowledge bases. Recommendation strategies could be different depending on which knowledge representation model and which order of the navigation models are used.

Both navigation models which are WPNavNet and TermNavNet enable to make Web-page recommendations. WPNavNet can directly infer next pages given the last one or two visited pages. TermNavNet is just able to predict next domain terms given the last one or two viewed domain terms, so, to have next pages recommended, the domain knowledge representation models, i.e. DomainOntoWP or TermNetWP, must be involved to query for domain terms of the visited Web-pages, and the Web-pages of the predicted next domain terms. A preliminary observation was performed to assess both navigation models; the TermNavNet-based recommendation model does not perform as good as the WPNavNet-based recommendation model, because a domain term in TermNavNet only presents part of the entire meaning of a Web-page. In addition, the process speed of WPNavNet is faster than that of TermNavNet because WPNavNet does not take time to query for domain terms and Web-pages. However, a coordination of both navigation models can improve the performance of Web-page recommendation since they can support each other, namely TermNavNet can overcome the "new-page" problem of WPNavNet. Depending on the strategy of using TermNavNet, the coordination can achieve differently high performance of Web-page recommendation. Therefore, WPNavNet and the coordination of WPNavNet and TermNavNet are considered in the Web-page recommendation strategies.

Besides, the navigation models predict a next state based on the first or second-order transition probabilities, so making Web-page recommendation can be based on the first or second-order WPNavNet, or the first or second-order TermNavNet, respectively. When we consider the currently accessed page to predict the next page to be accessed by a user, the first-order navigation models should be applied. Similarly, when we consider the last two

accessed pages to predict the next page, the second-order navigation models should be applied. Hence, if TermNavNet is coordinated with WPNavNet, both models should be performed on the same order.

From the above analysis, fourteen various Web-page recommendation strategies are designed and classified into three types, as follows.

Type 1: Web-page recommendation without semantic enhancement

- Two strategies apply the weighted networks of Web-page navigation, namely the first-order WPNavNet and the second-order WPNavNet, respectively, to Web-page recommendation. The two strategies are referred to as **R.WP.1st** and **R.WP.2nd** with respective to the first and second-order WPNavNet.

Type 2: Web-page recommendation with semantic enhancement based on the domain ontology

- Two strategies coordinate the first or second-order WPNavNets respectively with the weighted networks of domain term navigation, namely the first-order TermNavNet or second-order TermNavNet, integrated with the domain ontology, namely DomainOntoWP. Firstly, the WPNavNet is used to predict some next pages. Secondly, DomainOntoWP is used to query for domain terms of the last visited Web-pages. Thirdly, the TermNavNet is used to predict domain terms, and then the DomainOntoWP is used to query more Web-pages mapped to *each* of the predicted domain terms. The set of resulting Web-pages are recommended to an active user. The two strategies are referred to as **R.WP.ManTopic.1st.1** and **R.WP.ManTopic.2nd.1** with respective to the used first and second-order models.

Type 3: Web-page recommendation with semantic enhancement based on the semantic network

- Two strategies coordinate the first or second-order WPNavNets respectively with the first or second-order TermNavNets, integrated with the semantic network, namely TermNetWP. These two strategies perform as similarly as strategies

R.WP.ManTopic.1st.1 and **R.WP.ManTopic.2nd.1**, respectively, but TermNetWP is used instead of DomainOntoWP. The two strategies are referred to as **R.WP.AutoTopic.1st.1** and **R.WP.AutoTopic.2nd.1** with respective to the used first and second-order models.

- Two strategies coordinate the first or second-order WPNavNets respectively with the first or second-order TermNavNets, integrated with the TermNetWP. Firstly, the WPNavNet is used to predict some next pages. Secondly, TermNetWP is used to query for domain terms of the last visited Web-pages. Thirdly, the TermNavNet is used to predict domain terms, and then the TermNetWP is used to query more Web-pages mapped to *each group* of the predicted domain terms corresponding to each currently viewed domain term or each pair of currently and previously viewed domain terms. The set of resulting Web-pages are recommended to an active user. The two strategies are referred to as **R.WP.AutoTopic.1st.2** and **R.WP.AutoTopic.2nd.2** with respective to the used first and second-order models.

- Two strategies coordinate the first or second-order WPNavNets respectively with the first or second-order TermNavNets, integrated with the TermNetWP. Firstly, the WPNavNet is used to predict some next pages. Secondly, TermNetWP is used to query for domain terms of the last visited Web-pages. Thirdly, the TermNavNet is used to predict domain terms, and then the TermNetWP is used to query more Web-pages mapped to *all* of the predicted domain terms. The set of resulting Web-pages are recommended to an active user. The two strategies are referred to as **R.WP.AutoTopic.1st.3** and **R.WP.AutoTopic.2nd.3** with respective to the used first and second-order models.

- Two strategies coordinate the first or second-order WPNavNets respectively with the first or second-order TermNavNets, integrated with the TermNetWP. Firstly, the WPNavNet is used to predict some next pages. These two strategies are similar to strategies **R.WP.AutoTopic.1st.2** and **R.WP.AutoTopic.2nd.2**, respectively, but domain term prediction probabilities are considered to query Web-pages for recommendation. The two strategies are referred to as **R.WP.AutoTopic.1st.4** and

R.WP.AutoTopic.2nd.4 with respective to the used first and second-order models.

- Two strategies coordinate the first or second-order WPNavNets respectively with the first or second-order TermNavNets, integrated with the TermNetWP. Firstly, the WPNavNet is used to make Web-page recommendation. These two strategies are similar to strategies **R.WP.AutoTopic.1st.3** and **R.WP.AutoTopic.2nd.3**, respectively, but domain term prediction probabilities are considered to query Web-pages for recommendation. The two strategies are referred to as **R.WP.AutoTopic.1st.5** and **R.WP.AutoTopic.2nd.5** with respective to the used first and second-order models.

Table 7-2 summaries the fourteen Web-page recommendation strategies along with used models and reasoning algorithms. For each strategy, the models representing domain and Web usage knowledge can be used for making Web-page recommendations. Reasoning algorithms can be used to query for next pages, the domain terms of a Web-page, next domain terms, and Web-pages mapping to domain terms.

Table 7-2: Web-page recommendation strategies

Strategy	Recommend	Used models		Used reasoning algorithms			
		Domain	Web usage	Pred.Page	Q.Term	Pred.Term	Q.Page
R.WP.1st	the predicted next pages given a current page	-	1st-order WPNavNet	Alg.6.3 – RecPage (curP, O)			
R.WP.2nd	the predicted next pages given a current page and a previous page	-	2nd-order WPNavNet	Alg.6.2 – RecPage (preP, curP, O)			
R.WP.ManTopic.1st.1	the predicted next pages given a current page; and the pages of each predicted next domain term given each current domain term	Domain OntoWP	1st-order WPNavNet, 1st-order TermNavNet	Alg.6.3 – RecPage (curP, O)	Alg.4.2 – $Topic_{man}$ (d)	Alg.6.6 – RecDTerm (curT, O_T)	Alg.4.3 – $Page_{man}$(t)

R.WP.ManTopic.2nd.1	the predicted next pages given a current page and a previous page; and the pages of each predicted next domain term given each current domain term and previous domain term	Domain OntoWP	2nd-order WPNavNet, 2nd-order TermNavNet	Alg.6.2 – RecPage (preP, curP, O)	Alg.4.2 – $Topic_{man}$ (d)	Alg.6.5 – RecDTerm (preT, curT, O_T)	Alg.4.3 – $Page_{man}$(t)
R.WP.AutoTopic.1st.1	the predicted next pages given a current page; and the pages of each predicted next domain term given each current domain term	TermNet WP	1st-order WPNavNet, 1st-order TermNavNet	Alg.6.3 – RecPage (curP, O)	Alg.5.3 – $Topic_{auto}$ (d)	Alg.6.6 – RecDTerm (curT, O_T)	Alg.5.4 – $Page_{auto}$(t)
R.WP.AutoTopic.2nd.1	the predicted next pages given a current page and a previous page; and the pages of each predicted next domain term given each current domain term and previous domain term	TermNet WP	2nd-order WPNavNet, 2nd-order TermNavNet	Alg.6.2 – RecPage (preP, curP, O)	Alg.5.3 – $Topic_{auto}$ (d)	Alg.6.5 – RecDTerm (preT, curT, O_T)	Alg.5.4 – $Page_{auto}$(t)
R.WP.AutoTopic.1st.2	the predicted next pages given a current page; and the pages of predicted next domain terms given EACH current	TermNet WP	1st-order WPNavNet, 1st-order TermNavNet	Alg.6.3 – RecPage (curP, O)	Alg.5.3 – $Topic_{auto}$ (d)	Alg.6.6 – RecDTerm (curT, O_T)	Alg.5.5 – $Page_{auto}$ (τ)

	domain term						
R.WP.AutoTopic.2nd.2	the predicted next pages given a current page and a previous page; and the pages of predicted next domain terms given EACH pair of current and previous domain terms	TermNetWP	2nd-order WPNavNet, 2nd-order TermNavNet	Alg.6.2 – RecPage (preP, curP, O)	Alg.5.3 – $Topic_{auto}$ (d)	Alg.6.5 – RecDTerm (preT, curT, O_T)	Alg.5.5 – $Page_{auto}$ (τ)
R.WP.AutoTopic.1st.3	the predicted next pages given a current page; and the pages of ALL predicted next domain terms given each current domain terms	TermNetWP	1st-order WPNavNet, 1st-order TermNavNet	Alg.6.3 – RecPage (curP, O)	Alg.5.3 – $Topic_{auto}$ (d)	Alg.6.6 – RecDTerm (curT, O_T)	Alg.5.5 – $Page_{auto}$ (τ)
R.WP.AutoTopic.2nd.3	the predicted next pages given a current page and a previous page; and the pages of ALL predicted next domain terms given each pair of current and previous domain terms	TermNetWP	2nd-order WPNavNet, 2nd-order TermNavNet	Alg.6.2 – RecPage (preP, curP, O)	Alg.5.3 – $Topic_{auto}$ (d)	Alg.6.5 – RecDTerm (preT, curT, O_T)	Alg.5.5 – $Page_{auto}$ (τ)
R.WP.AutoTopic.1st.4	the predicted next pages given a current page; and the pages of predicted next domain	TermNetWP	1st-order WPNavNet, 1st-order TermNavNet	Alg.6.3 – RecPage (curP, O)	Alg.5.3 – $Topic_{auto}$ (d)	Alg.6.8 – RecDTermProb (curT, O_T)	Alg.5.6 – PageProb (τ')

	terms with estimated probabilities given EACH current domain term						
R.WP.AutoTopic.2nd.4	the predicted next pages given a current page and a previous page; and the pages of predicted next domain terms with estimated probabilities given EACH pair of current and previous domain terms	TermNetWP	2nd-order WPNavNet, 2nd-order TermNavNet	Alg.6.2 – RecPage (preP, curP, O)	Alg.5.3 – $Topic_{auto}$ (d)	Alg.6.7 – RecDTermProb (preT, curT, O_T)	Alg.5.6 – PageProb (τ′)
R.WP.AutoTopic.1st.5	the predicted next pages given a current page; and the pages of ALL predicted next domain terms with estimated probabilities	TermNetWP	1st-order WPNavNet, 1st-order TermNavNet	Alg.6.3 – RecPage (curP, O)	Alg.5.3 – $Topic_{auto}$ (d)	Alg.6.8 – RecDTermProb (curT, O_T)	Alg.5.6 – PageProb (τ′)
R.WP.AutoTopic.2nd.5	the predicted next pages given a current page and a previous page; and the pages of ALL predicted next domain terms with estimated probabilities	TermNetWP	2nd-order WPNavNet, 2nd-order TermNavNet	Alg.6.2 – RecPage (preP, curP, O)	Alg.5.3 – $Topic_{auto}$ (d)	Alg.6.7 – RecDTermProb (preT, curT, O_T)	Alg.5.6 – PageProb (τ′)

Definition 7.1 Based on Definitions 6.1, and 6.7, let $S = s_1 ... s_k$ ($s_i \in D$, $i = [1..k]$) be a

sequence of Web-pages, which have been visited by a user, with s_k being the currently accessed page and s_{k-1} being the previously accessed page; $P = d_1 ... d_k ... d_m \in \mathbb{P}$ ($d_j \in D, j = [1..m]$) be a pattern of Web-pages in FWAP modelled in WPNavNet, with d_{k-1} and d_k being Web-pages matching with s_{k-1} and s_k, respectively, and d_{k+1} being a predicted next page; $F = t_1 ... t_k ... t_m \in \mathbb{F}$ ($t_j \in \mathbb{T}, j = [1..m]$) be a pattern of domain terms in FVTP modelled in TermNavNet with t_k being a currently viewed domain term of page s_k, t_{k-1} being a previously viewed domain term of page s_{k-1}, and t_{k+1} being a predicted next domain term.

Based on Definition 7.1, the following presents the details of the Web-page recommendation strategies step by step. A recommendation length N is required to make sure that each strategy recommends the top-N predicted next Web-pages.

For strategy R.WP.1st, the steps are carried out as follows:

Step 1 generates FWAP using PLWAP-Mine;

Step 2 builds a 1st-order WPNavNet given FWAP; and

Step 3 infers next visited pages { d_{k+1} } given a page d_k using Algorithm 6-3 based on the 1st-order WPNavNet.

For strategy R.WP.2nd, the steps are carried out as follows:

Step 1 generates FWAP using PLWAP-Mine;

Step 2 builds a 2nd-order WPNavNet given FWAP; and

Step 3 infers next visited pages { d_{k+1} } given a pair pages (d_{k-1}, d_k) using Algorithm 6-2 based on the 2nd-order WPNavNet.

Based on Algorithms 6-2 and 6-3, a procedure recommending Web-pages in Step 3 for two strategies R.WP.1st and R.WP.2nd, respectively, is listed in Algorithm 7-1.

Algorithm 7-1: Web-page recommendation without semantic enhancement

Input:
preP (previous page) (*)
curP (current page)
O_1 (WPNavNet)
Output: *recP* (recommended pages)
Process:
Set *recP* = ***RecPage***(*preP*, *curP*, O_1)
Return *recP*

Notes:
- (*) For R.WP.1st, *preP* is not involved.

For two strategies R.WP.ManTopic.1st.1 and R.WP.ManTopic.2nd.1, the steps are carried out as follows:

Step 1 builds the domain knowledge of Web-pages at a given website using DomainOntoWP;

Step 2 generates FWAP using PLWAP-Mine;

Step 3 builds a 1st or 2nd-order WPNavNet for the two strategies, respectively, given FWAP;

Step 4 builds FVTP;

Step 5 builds a 1st or 2nd-order TermNavNet for the two strategies, respectively, given FVTP;

Step 6 infers next visited pages $\{ d_{k+1} \}$ given a page d_k using Algorithm 6-3 if the 1st-order WPNavNet is applied; or given a pair pages (d_{k-1}, d_k) using Algorithm 6-2 if the 2nd-order WPNavNet is applied;

If no next page is predicted, or the number of next pages is fewer than N, then the next steps are taken into account;

Step 7 identifies a set of previously viewed domain terms $\{ t_{k-1} \}$ for the second strategy only, and a set of currently viewed domain terms $\{ t_k \}$ using Algorithm 4-2 based on DomainOntoWP;

Step 8 infers next viewed domain terms $\{ t_{k+1} \}$ given each domain term in $\{ t_k \}$ using Algorithm 6-6 if the 1st-order TermNavNet is applied; or given each pair domain terms in $\{ t_{k-1}, t_k \}$ using Algorithm 6-5 if the 2nd-order TermNavNet is applied; and

Step 9 recommends the predicted next Web-pages in Step 6, and Web-pages mapped to each domain term in $\{ t_{k+1} \}$ using Algorithm 4-3 based on DomainOntoWP.

Based on the description of Algorithms 4-2, 4-3, 6-2, 6-3, 6-5, and 6-6, two strategies R.WP.ManTopic.1st.1 and R.WP.ManTopic.2nd.1, referred to as $R.WP.ManTopic_1(d_k)$ and $R.WP.ManTopic_1(d_{k-1}, d_k)$, respectively, is described in logics notation as shown in Table 7-3.

Table 7-3: Description logics notation of strategies R.WP.ManTopic.1st.1 and R.WP.ManTopic.2nd.1

R.WP.ManTopic.1st.1	R.WP.ManTopic.2nd.1
$R.WP.ManTopic_1(d_k) = RecPage(d_k) \cup \bigcup_{\substack{t_z \in T_z \\ 1 \le z \le \lvert T_z \rvert}} Page_{man}(t_z)$, where $T_x = Topic_{man}(d_k)$, $T_z = \bigcup_{\substack{t_x \in T_x \\ 1 \le x \le \lvert T_x \rvert}} RecDTerm(t_x)$.	$R.WP.ManTopic_1(d_{k-1}, d_k) = RecPage(d_{k-1}, d_k) \cup \bigcup_{\substack{t_z \in T_z \\ 1 \le z \le \lvert T_z \rvert}} Page_{man}(t_z)$, where $T_x = Topic_{man}(d_{k-1})$, $T_y = Topic_{man}(d_k)$, $T_z = \bigcup_{\substack{t_y \in T_y \\ 1 \le y \le \lvert T_y \rvert}} \bigcup_{\substack{t_x \in T_x \\ 1 \le x \le \lvert T_x \rvert}} RecDTerm(t_x, t_y)$.

A procedure from Step 6 to Step 9 for the two strategies is listed in Algorithm 7-2.

Algorithm 7-2: Web-page recommendation with semantic enhancement based on DomainOntoWP

Input:
  preP (previous page) (*)
  curP (current page)
  O_1 (WPNavNet)
  O_2 (TermNavNet)
  O_3 (DomainOntoWP)
  N (the recommendation length)
Output: *recP* (recommended pages)
Process:
  Set *recP* = ***RecPage***(*preP*, *curP*, O_1)
If (*recP* = *null* or |*recP*| < *N*) {
  Traverse the ontology O_3 to:
    Set *curT* = ***Topic**man* (*curP*)
    Set *preT* = ***Topic**man* (*preP*)
  For each $curT_i$ in *curT*,
    For each $preT_j$ in *preT*,
      Set *recT* = ***RecDTerm***($preT_j$, $curT_i$, O_2)
      For each *recT_k* in *recT*,
        Set *P* = ***Page**man*(*recT_k*)
        Add *P* into *recP*
}
  Return *recP*

Notes:
- (*) For R.WP.ManTopic.1st.1, *preP* is not involved.

For two strategies R.WP.AutoTopic.1st.1 and R.WP.AutoTopic.2nd.1, the steps are carried out as follows:

Step 1 builds the domain knowledge of Web-pages at a given website using TermNetWP;

Steps 2-6 are similar to the ones specified for the two above strategies;

If no next page is predicted, or the number of next pages is fewer than *N*, then the next steps are taken into account;

Step 7 identifies a set of previously viewed domain terms { t_{k-1} } for the second strategy only, and a set of currently viewed domain terms { t_k } using Algorithm 5-3 based on TermNetWP;

Step 8 similar to the one specified for the two above strategies; and

Step 9 recommends the predicted next Web-pages in Step 6, and Web-pages mapped to each domain term in { t_{k+1} } using Algorithm 5-4 based on TermNetWP.

Based on the description of Algorithms 5-3, 5-4, 6-2, 6-3, 6-5, and 6-6, two strategies R.WP.AutoTopic.1st.1 and R.WP.AutoTopic.2nd.1, referred to as $R.WP.AutoTopic_1(d_k)$ and $R.WP.AutoTopic_1(d_{k-1}, d_k)$, respectively, is described in logics notation as shown in Table 7-4.

Table 7-4: Description logics notation of strategies R.WP.AutoTopic.1st.1 and R.WP.AutoTopic.2nd.1

R.WP.AutoTopic.1st.1	R.WP.AutoTopic.2nd.1
$R.WP.AutoTopic_1(d_k) = RecPage(d_k) \cup \bigcup_{\substack{t_z \in T_z \\ 1 \le z \le \lvert T_z \rvert}} Page_{auto}(t_z)$, where $T_x = Topic_{auto}(d_k)$, $T_z = \bigcup_{\substack{t_x \in T_x \\ 1 \le x \le \lvert T_x \rvert}} RecDTerm(t_x)$.	$R.WP.AutoTopic_1(d_{k-1}, d_k) = RecPage(d_{k-1}, d_k) \cup \bigcup_{\substack{t_z \in T_z \\ 1 \le z \le \lvert T_z \rvert}} Page_{auto}(t_z)$, where $T_x = Topic_{auto}(d_{k-1})$, $T_y = Topic_{auto}(d_k)$, $T_z = \bigcup_{\substack{t_y \in T_y \\ 1 \le y \le \lvert T_y \rvert}} \bigcup_{\substack{t_x \in T_x \\ 1 \le x \le \lvert T_x \rvert}} RecDTerm(t_x, t_y)$.

A procedure from Step 6 to Step 9 for the two strategies is listed in Algorithm 7-3.

Algorithm 7-3: Web-page recommendation with semantic enhancement based on TermNetWP

Input:
preP (previous page) (*)
curP (current page)
O_1 (WPNavNet)
O_2 (TermNavNet)
O_3 (TermNetWP)
N (the recommendation length)
Output: *recP* (recommended pages)
Process:
Set *recP* = ***RecPage***(*preP*, *curP*, O_1)
If (*recP* = *null* or |*recP*| < *N*) {
Traverse the ontology O_3 to:
Set *curT* = $\boldsymbol{Topic}_{auto}$(*curP*)
Set *preT* = $\boldsymbol{Topic}_{auto}$(*preP*)
For each $curT_i$ in *curT*,
For each $preT_j$ in *preT*,
Set *recT* = ***RecDTerm***($preT_j$, $curT_i$, O_2)
For each *recT_k* in *recT*,
Set *P* = $\boldsymbol{Page}_{auto}$(*recT_k*)
Add *P* into *recP*
}
Return *recP*

Notes:
- (*) For R.WP.AutoTopic.1st.1, *preP* is not involved.

For two strategies R.WP.AutoTopic.1st.2 and R.WP.AutoTopic.2nd.2, the steps are carried out as follows:

Steps 1-6 are similar to the ones specified for strategies R.WP.AutoTopic.1st.1 and R.WP.AutoTopic.2nd.1;

If no next page is predicted, or the number of next pages is fewer than *N*, then the next steps are taken into account;

Step 7 is similar to the one specified for strategies R.WP.AutoTopic.1st.1 and R.WP.AutoTopic.2nd.1;

Step 8 infers next viewed domain terms { t_{k+1} } given EACH domain term in { t_k } using Algorithm 6-6 if the 1st-order TermNavNet is applied; or given EACH pair domain terms in { t_{k-1}, t_k } using Algorithm 6-5 if the 2nd-order TermNavNet is applied; and

Step 9 recommends the predicted next Web-pages in Step 6, and Web-pages mapped to EACH { t_{k+1} } using Algorithm 5-5 based on TermNetWP.

Based on the description of Algorithms 5-3, 5-5, 6-2, 6-3, 6-5, and 6-6, two strategies R.WP.AutoTopic.1st.2 and R.WP.AutoTopic.2nd.2, referred to as $R.WP.AutoTopic_2(d_k)$

and $R.WP.AutoTopic_2(d_{k-1}, d_k)$, respectively, is described in logics notation as shown in Table 7-5.

Table 7-5: Description logics notation of strategies R.WP.AutoTopic.1st.2 and R.WP.AutoTopic.2nd.2

R.WP.AutoTopic.1st.2	R.WP.AutoTopic.2nd.2
$R.WP.AutoTopic_2(d_k) = RecPage(d_k) \cup \bigcup_{1 \le x \le \lvert T_x \rvert} \bigcup_{t_x \in T_x} Page_{auto}(RecDTerm(t_x))$, where $T_x = Topic_{auto}(d_k)$.	$R.WP.AutoTopic_2(d_{k-1}, d_k) = RecPage(d_{k-1}, d_k) \cup \bigcup_{1 \le y \le \lvert T_y \rvert} \bigcup_{t_y \in T_y} \bigcup_{1 \le x \le \lvert T_x \rvert} \bigcup_{t_x \in T_x} Page_{auto}(RecDTerm(t_x, t_y))$, where $T_x = Topic_{auto}(d_{k-1})$, $T_y = Topic_{auto}(d_k)$.

A procedure from Step 6 to Step 9 for the two strategies is listed in Algorithm 7-4.

Algorithm 7-4: Web-page recommendation with semantic enhancement based on TermNetWP

Input:
  preP (previous page) (*)
  curP (current page)
  O_1 (WPNavNet)
  O_2 (TermNavNet)
  O_3 (TermNetWP)
  N (the recommendation length)
Output: *recP* (recommended pages)
Process:
  Set *recP* = ***RecPage***(*preP*, *curP*, O_1)
If (*recP* = *null* or |*recP*| < *N*) {
  Traverse the ontology O_3 to:
    Set *curT* = ***Topic***$_{auto}$(*curP*)
    Set *preT* = ***Topic***$_{auto}$(*preP*)
  For each $curT_i$ in *curT*,
    For each $preT_j$ in *preT*,
      Set *recT* = ***RecDTerm***($preT_j$, $curT_i$, O_2)
      Set *P* = ***Page***$_{auto}$(*recT*)
      Add *P* into *recP*
}
  Return *recP*

Notes:
- (*) For R.WP.AutoTopic.1st.2, *preP* is not involved.

For two strategies R.WP.AutoTopic.1st.3 and R.WP.AutoTopic.2nd.3, the steps are carried out as follows:

Steps 1-6 are similar to the ones specified for strategies R.WP.AutoTopic.1st.1 and R.WP.AutoTopic.2nd.1;

If no next page is predicted, or the number of next pages is fewer than *N*, then the next

steps are taken into account;

Step 7 is similar to the one specified for strategies R.WP.AutoTopic.1st.1 and R.WP.AutoTopic.2nd.1;

Step 8 infers next viewed domain terms $\{ t_{k+1} \}$ given each domain term in $\{ t_k \}$ using Algorithm 6-6 if the 1st-order TermNavNet is applied; or given each pair domain terms in $\{ t_{k-1}, t_k \}$ using Algorithm 6-5 if the 2nd-order TermNavNet is applied; and results in a set of next domain terms $\{\{ t_{k+1} \}\}$;

Step 9 recommends the predicted next Web-pages in Step 6, and Web-pages mapped to the SET $\{\{ t_{k+1} \}\}$ using Algorithm 5-5 based on TermNetWP.

Based on the description of Algorithms 5-3, 5-5, 6-2, 6-3, 6-5, and 6-6, two strategies R.WP.AutoTopic.1st.3 and R.WP.AutoTopic.2nd.3, referred to as $R.WP.AutoTopic_3(d_k)$ and $R.WP.AutoTopic_3(d_{k-1}, d_k)$, respectively, is described in logics notation as shown in Table 7-6.

Table 7-6: Description logics notation of strategies R.WP.AutoTopic.1st.3 and R.WP.AutoTopic.2nd.3

R.WP.AutoTopic.1st.3	R.WP.AutoTopic.2nd.3
$R.WP.AutoTopic_3(d_k) = RecPage(d_k) \cup Page_{auto}(T_z)$, where $T_x = Topic_{auto}(d_k)$, $T_z = \bigcup_{\substack{t_x \in T_x \\ 1 \le x \le \lvert T_x \rvert}} RecDTerm(t_x)$.	$R.WP.AutoTopic_3(d_{k-1}, d_k) = RecPage(d_{k-1}, d_k) \cup Page_{auto}(T_z)$, where $T_x = Topic_{auto}(d_{k-1})$, $T_y = Topic_{auto}(d_k)$, $T_z = \bigcup_{\substack{t_y \in T_y \\ 1 \le y \le \lvert T_y \rvert}} \bigcup_{\substack{t_x \in T_x \\ 1 \le x \le \lvert T_x \rvert}} RecDTerm(t_x, t_y)$.

A procedure from Step 6 to Step 9 for the two strategies is listed in Algorithm 7-5.

Algorithm 7-5: Web-page recommendation with semantic enhancement based on TermNetWP

Input:
  preP (previous page) (*)
  curP (current page)
  O_1 (WPNavNet)
  O_2 (TermNavNet)
  O_3 (TermNetWP)
  N the recommendation length
Output: *recP* (recommended pages)
Process:
  Set *recP* = ***RecPage***(*preP*, *curP*, O_1)
If (*recP* = *null* or |*recP*| < *N*) {
  Traverse the ontology O_3 to:
    Set *curT* = $\boldsymbol{Topic_{auto}}$(*curP*)
    Set *preT* = $\boldsymbol{Topic_{auto}}$(*preP*)
  Set *recT* = null
  For each $curT_i$ in *curT*,
    For each $preT_j$ in *preT*,
      Set *recT_k* = ***RecDTerm***($preT_j$, $curT_i$, O_2)
      Add *recT_k* into *recT*
  Set *P* = $\boldsymbol{Page_{auto}}$(*recT*)
  Add *P* into *recP*
}
  Return *recP*

Notes:
- (*) For R.WP.AutoTopic.1st.3, *preP* is not involved.

For two strategies R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.2nd.4, the steps are carried out as follows:

Steps 1-6 are similar to the ones specified for strategies R.WP.AutoTopic.1st.1 and R.WP.AutoTopic.2nd.1;

If no next page is predicted, or the number of next pages is fewer than *N*, then the next steps are taken into account;

Step 7 is similar to the one specified for strategies R.WP.AutoTopic.1st.1 and R.WP.AutoTopic.2nd.1;

Step 8 infers next viewed domain terms { t_{k+1} } with respective prediction probabilities given EACH domain term in { t_k } using Algorithm 6-8 if the 1st-order TermNavNet is applied; or given EACH pair domain terms in { t_{k-1}, t_k } using Algorithm 6-7 if the 2nd-order TermNavNet is applied; and

Step 9 recommends the predicted next Web-pages in Step 6, and Web-pages mapped to EACH { t_{k+1} } using Algorithm 5-6 based on TermNetWP.

Based on the description of Algorithms 5-3, 5-6, 6-2, 6-3, 6-7, and 6-8, two strategies R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.2nd.4, referred to as $R.WP.AutoTopic_4(d_k)$ and $R.WP.AutoTopic_4(d_{k-1}, d_k)$, respectively, is described in logics notation as shown in Table 7-7.

Table 7-7: Description logics notation of strategies R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.2nd.4

R.WP.AutoTopic.1st.4	R.WP.AutoTopic.2nd.4
$R.WP.AutoTopic_4(d_k) = RecPage(d_k) \cup \bigcup_{\substack{t_x \in T_x \\ 1 \le x \le \lvert T_x \rvert}} PageProb(RecDTermProb(t_x))$, where $T_x = Topic_{auto}(d_k)$.	$R.WP.AutoTopic_4(d_{k-1}, d_k) = RecPage(d_{k-1}, d_k) \cup \bigcup_{\substack{t_y \in T_y \\ 1 \le y \le \lvert T_y \rvert}} \bigcup_{\substack{t_x \in T_x \\ 1 \le x \le \lvert T_x \rvert}} PageProb(RecDTermProb(t_x, t_y))$, where $T_x = Topic_{auto}(d_{k-1})$, $T_y = Topic_{auto}(d_k)$.

A procedure from Step 6 to Step 9 for the two strategies is listed in Algorithm 7-6.

Algorithm 7-6: Web-page recommendation with semantic enhancement based on TermNetWP

Input:
  preP (previous page) (*)
  curP (current page)
  O_1 (WPNavNet)
  O_2 (TermNavNet)
  O_3 (TermNetWP)
  N the recommendation length
Output: *recP* (recommended pages)
Process:
  Set *recP* = ***RecPage***(*preP*, *curP*, O_1)
If (*recP* = *null* or |*recP*| < *N*) {
  Traverse the ontology O_3 to:
    Set *curT* = ***Topic***$_{auto}$(*curP*)
    Set *preT* = ***Topic***$_{auto}$(*preP*)
  For each $curT_i$ in *curT*,
    For each $preT_j$ in *preT*,
      Set *recT_Prob* = ***RecDTermProb***($preT_j$, $curT_i$, O_2)
      Set *P* = ***PageProb***(*recT_Prob)*
      Add *P* into *recP*
}
  Return *recP*

Notes:
- (*) For R.WP.AutoTopic.1st.4, *preP* is not involved.

For two strategies R.WP.AutoTopic.1st.5 and R.WP.AutoTopic.2nd.5, the steps are carried out as follows:

Steps 1-6 are similar to the ones specified for strategies R.WP.AutoTopic.1st.1 and R.WP.AutoTopic.2nd.1;

If no next page is predicted, or the number of next pages is fewer than *N*, then the next steps are taken into account;

Step 7 is similar to the one specified for strategies R.WP.AutoTopic.1st.1 and R.WP.AutoTopic.2nd.1;

Step 8 infers next viewed domain terms { t_{k+1} } given each domain term in { t_k } using Algorithm 6-8 if the 1st-order TermNavNet is applied; or given each pair domain terms in { t_{k-1}, t_k } using Algorithm 6-7 if the 2nd-order TermNavNet is applied; and results in a set of next domain terms {{ t_{k+1} }} with respective prediction probabilities;

Step 9 recommends the predicted next Web-pages in Step 6, and Web-pages mapped to the SET {{ t_{k+1} }} using Algorithm 5-6 based on TermNetWP.

Based on the description of Algorithms 5-3, 5-6, 6-2, 6-3, 6-7, and 6-8, two strategies R.WP.AutoTopic.1st.5 and R.WP.AutoTopic.2nd.5, referred to as $R.WP.AutoTopic_5(d_k)$ and $R.WP.AutoTopic_5(d_{k-1}, d_k)$, respectively, is described in logics notation as shown in Table 7-8.

Table 7-8: Description logics notation of strategies R.WP.AutoTopic.1st.5 and R.WP.AutoTopic.2nd.5

R.WP.AutoTopic.1st.5	R.WP.AutoTopic.2nd.5
$R.WP.AutoTopic_5(d_k) = RecPage(d_k) \cup PageProb(T_z)$, where $T_x = Topic_{auto}(d_k)$, $T_z = \bigcup_{\substack{t_x \in T_x \\ 1 \le x \le \lvert T_x \rvert}} RecDTermProb(t_x)$.	$R.WP.AutoTopic_5(d_{k-1}, d_k) = RecPage(d_{k-1}, d_k) \cup PageProb(T_z)$, where $T_x = Topic_{auto}(d_{k-1})$, $T_y = Topic_{auto}(d_k)$, $T_z = \bigcup_{\substack{t_y \in T_y \\ 1 \le y \le \lvert T_y \rvert}} \bigcup_{\substack{t_x \in T_x \\ 1 \le x \le \lvert T_x \rvert}} RecDTermProb(t_x, t_y)$.

A procedure from Step 6 to Step 9 for the two strategies is listed in Algorithm 7-7.

Algorithm 7-7: Web-page recommendation with semantic enhancement based on TermNetWP

Input:
preP (previous page) (*)
curP (current page)
O_1 (WPNavNet)
O_2 (TermNavNet)
O_3 (TermNetWP)
N the recommendation length
Output: *recP* (recommended pages)
Process:
Set *recP* = ***RecPage***(*preP*, *curP*, O_1)
If (*recP* = *null* or |*recP*| < *N*) {
Traverse the ontology O_3 to:
Set *curT* = $\boldsymbol{Topic_{auto}}$(*curP*)
Set *preT* = $\boldsymbol{Topic_{auto}}$(*preP*)
Set *recT_Prob* = null
For each $curT_i$ in *curT*,
For each $preT_j$ in *preT*,
Set *recT_Prob _k* = ***RecDTermProb***($preT_j$, $curT_i$, O_2)
Add *recT_Prob_k* into *recT_Prob*
Set *P* = ***PageProb***(*recT_Prob*)
Add *P* into *recP*
}
Return *recP*

Notes:
- (*) For R.WP.AutoTopic.1st.5, *preP* is not involved.

Generally speaking, strategies **R.WP.1st** and **R.WP.2nd** are the Web-page recommendation strategies based on the transformed Web usage knowledge, i.e. WPNavNet, strategies **R.WP.ManTopic.1st.1**, **R.WP.ManTopic.2nd.1, R.WP.AutoTopic.1st.1** and **R.WP.AutoTopic.2nd.1** are considered as the basic semantically enhanced Web-page recommendation strategies, and the remaining strategies as the variations of **R.WP.AutoTopic.1st.1** and **R.WP.AutoTopic.2nd.1**.

7.4. Experimental Evaluation

In order to evaluate the effectiveness of the proposed models of knowledge representation and the recommendation strategies along with the set of reasoning algorithms, these models, algorithms and strategies are implemented to test their performance of Web-page recommendation using a public dataset (http://kdd.ics.uci.edu/databases/msweb/msweb.html) and a real world dataset

(handbook.uts.edu.au). This section firstly lists the measures for the performance evaluation of Web-page recommendation strategies, secondly presents a experimental method, and then describes experimental results with the public dataset and the real world dataset.

7.4.1. Performance Evaluation

The performance of Web-page recommendation strategies is measured in terms of two major performance metrics: *Precision* and *Satisfaction* according to Zhou (2004), introduced in Sub-section 3.5.1. The precision is useful to measure how probable a user will access one of the recommended Web-pages. Besides, we also need to consider if a user accesses one of the recommended Web-pages in the near future. Actually, the next page accessed by a user may not the target page that user wants. In many cases, a user has to access a few intermediate pages before reaching the target page. Hence, the satisfaction is necessary to give the precision that the recommended pages will be accessed in the near future. In order to calculate these two metrics, this sub-section first introduces a definition of Web-page recommendation rules in a similar way as Definition 3.2.

Definition 7.2 *(Web-page recommendation rules)*

Let $S = s_1 s_2 \ldots s_k\, s_{k+1} \ldots s_n$ be a WAS. For each prefix sequence $S_{prefix} = s_1\, s_2 \ldots s_k\, (1 \le k \le n\text{-}1)$, a Web-page recommendation rule is defined as a set of recommended Web-pages which are generated by a Web-page recommendation strategy, denoted as $RR = \{r_1, r_2, \ldots, r_M\}$, where $r_i\, (i = [1..M])$ is a recommended Web-page.

A Web-page recommendation rule is deemed as *correct*, and/or *satisfied*, or *empty* based on the following conditions:

- If $s_{k+1} \in RR$, RR is *correct*.
- If $\exists s_i \epsilon RR\ (k+1 \le i \le n)$, RR is *satisfied*.
- If $M = 0$, RR is *empty*.

Given a set of recommendation rules, $R = \{RR_1, RR_2 \ldots RR_N\}$, where $RR_i\, (1 \le i \le N)$ is a recommendation rule, and $|R| = N$ is the total number of recommendation rules in R including *empty* rules. The Precision and Satisfaction can be defined as Definitions 3.3 and 3.4.

7.4.2. Experimental Method

An algorithm, as Algorithm 7-8, is developed to calculate the performance evaluation measures for all experiments in the similar way as Algorithm 3-5 in Chapter 3. The strategies proposed in Section 7-3 are used to generate Web-page recommendation rules. Note that infrequent items are not removed from the testing WAS like Algorithm 3-5, because the purpose of the all experiments in this section is to examine how the applied models can overcome this challenge.

Algorithm 7-8: Performance evaluation

Input:
WASD (Web access sequence dataset)
MinSup (the minimum support threshold)
N (the recommendation length)
O_1 (WPNavNet)
O_2 (TermNavNet)
O_3 (DomainOntoWP) (or TermNetWP)
Output:
Precision
Satisfaction
Process:
For each sequence S_i in *WASD*:
$S_i = s_1 s_2 \ldots s_k s_{k+1} \ldots s_n$
For each $k \in [1..n-1]$:
Set $curP = s_k$
Set $preP = s_{k-1}$
Given the parameters ($curP$, $preP$, O_1, O_2, O_3), generate the top-N recommended pages $recP = \{r_1, r_2, \ldots, r_M\}$ using one Web-page recommendation strategy with respect to a certain experimental case
If $s_{k+1} \in recP$, then increase the number of *correct* recommendation rules $|R_c|$ by 1
If $\exists s_i \in recP\ (k+1 \le i \le n)$, then increase the number of *satisfied* recommendation rules $|R_s|$ by 1
Increase the number of recommendation rules $|R|$
Return: the *precision* and *satisfaction*, computed using Equations (3.3) and (3.4), respectively

Note that in case PLWAP-Mine is used to make Web-page recommendations, Algorithm 3-5 is used for its performance evaluation (referring to Chapter 3), but infrequent items are not removed from sequence S_i.

Two important parameters, namely MinSup, which is the threshold of support values for a Web access sequence to be qualified as a frequent Web access pattern, and recommendation length, which is the number of recommended next Web-pages, might have significant impact on the performance of Web-page recommendation. The MinSup values are chosen depending on the training dataset to guarantee that the amount of generated information is enough to conduct the proposed knowledge representation models and make Web-page recommendations. In other words, when the MinSup value is too small, the size

of discovered FWAP will be too large to run the algorithms, and when the MinSup value is too high, the size of FWAP will become too small to predict Web-pages. Therefore, it is necessary to observe the training dataset first to determine the range of appropriate MinSup values. Regarding the recommendation length, it is pre-defined to make the system only provide the top-*N* recommended Web-pages that have higher prediction probabilities. The recommended pages with lower prediction probabilities are ignored to speed the recommendation processes. The recommendation length is limited to the typical value 5 for each experimental case in Sub-sections 7.4.3 and 7.4.4.

All experiments are implemented in Java in conjunction with Protégé based on the new models, algorithms and strategies, and the competing recommendation model using PLWAP-Mine, and run in an Intel Core i5-460M, 2.53 GHz Windows 7 machine with 4GB of RAM.

7.4.3. Experiments with Public Datasets

Training and Testing Datasets

In this section, the Web usage dataset of the Microsoft (MS) website (*www.microsoft.com*) was used to run the experimental cases. The dataset was downloaded from http://kdd.ics.uci.edu/databases/msweb/msweb.html. This dataset records which areas (Vroots) of www.microsoft.com each user visited in a one-week timeframe in Feburary 1998. In the dataset, users are identified as numbers, and the 294 Web-pages (Vroots) are identified by their titles and URLs. Therefore, this dataset is suitable for the experiments. The dataset is divided into two sub-sets, one for training and one for testing. The two sub-sets are pre-processed into the format of WAS for mining. The training set has 32711 records, and the testing set has 5000 records. The average length of WAS in both sub-sets is 3.

Design of Experimental Cases

This sub-section compares the performance of different Web-page recommendation models ranging from the traditional model using PLWAP-Mine; through the Web-page recommendation models using the transformed Web usage knowledge (WPNavNet); to the

semantic-enhanced models based on a domain ontology of Web-pages (DomainOntoWP), which is constructed manually based on the developer's knowledge, and the DomainOntoWP-based semantic Web usage knowledge (TermNavNet); and the semantic-enhanced models based on a semantic network of Web-pages (TermNetWP), which is automatically constructed based on the Web usage dataset, and the TermNetWP-based semantic Web usage knowledge (TermNavNet). In addition, this section also takes a chance to compare how prediction accuracy can be achieved using the first and second-order navigation models, i.e. WPNavNet and TermNavNet, in experiment cases since comparing the accuracy of first and second-order prediction models, such as Markov models, has been a controversial issue.

Considering the new models of knowledge representation, the reasoning algorithms and the recommendation strategies, 15 experimental cases are designed and summarized in Table 7-9.

Table 7-9: Experimental cases

Case	Used models	Web-page recommendation strategy
1	PLWAP-Mine (Referring to Chapter 3)	R.PLWAP
2	1st order WPNavNet	R.WP.1st
3	2nd-order WPNavNet	R.WP.2nd
4	DomainOntoWP, 1st-order WPNavNet, 1st-order TermNavNet	R.WP.ManTopic.1st.1
5	DomainOntoWP, 2nd-order WPNavNet, 2nd-order TermNavNet	R.WP.ManTopic.2nd.1
6	TermNetWP, 1st-order WPNavNet, 1st-order TermNavNet	R.WP.AutoTopic.1st.1
7	TermNetWP, 2nd-order WPNavNet, 2nd-order TermNavNet	R.WP.AutoTopic.2nd.1
8	TermNetWP, 1st-order WPNavNet, 1st-order TermNavNet	R.WP.AutoTopic.1st.2
9	TermNetWP, 2nd-order WPNavNet, 2nd-order TermNavNet	R.WP.AutoTopic.2nd.2
10	TermNetWP, 1st-order WPNavNet, 1st-order TermNavNet	R.WP.AutoTopic.1st.3
11	TermNetWP, 2nd-order WPNavNet, 2nd-order TermNavNet	R.WP.AutoTopic.2nd.3
12	TermNetWP, 1st-order WPNavNet, 1st-order TermNavNet	R.WP.AutoTopic.1st.4
13	TermNetWP, 2nd-order WPNavNet, 2nd-order TermNavNet	R.WP.AutoTopic.2nd.4
14	TermNetWP, 1st-order WPNavNet, 1st-order TermNavNet	R.WP.AutoTopic.1st.5
15	TermNetWP, 2nd-order WPNavNet, 2nd-order TermNavNet	R.WP.AutoTopic.2nd.5

Case 1. (R.PLWAP) Set the threshold of the traditional Web-page recommendation model using a best Web usage mining (WUM) algorithm, i.e. PLWAP-Mine, as it has been proven more efficient than the other sequence mining algorithms in Chapter 3. In this case, the Web-page recommendations are generated based on the PLWAP-Mine algorithm. We refer to this case as the base case, namely R.PLWAP.

Case 2. (R.WP.1st) Test the effectiveness of the Web-page recommendation by using the first-order WPNavNet. The recommendation strategy (R.WP.1st) is used.

Case 3. (R.WP.2nd) Test the effectiveness of the Web-page recommendation by using the second-order WPNavNet. The recommendation strategy (R.WP.2nd) is used.

Case 4. (R.WP.ManTopic.1st.1) Test the effectiveness of the semantic-enhanced Web-page recommendation by coordinating the first-order WPNavNet, and the first-order TermNavNet integrated with the domain ontology (DomainOntoWP). The recommendation strategy (R.WP.ManTopic.1st.1) is used.

Case 5. (R.WP.ManTopic.2nd.1) Test the effectiveness of the semantic-enhanced Web-page recommendation by coordinating the second-order WPNavNet, and the second-order TermNavNet integrated with the domain ontology (DomainOntoWP). The recommendation strategy (R.WP.ManTopic.2nd.1) is used.

Case 6 (R.WP.AutoTopic.1st.1) Test the effectiveness of the semantic-enhanced Web-page recommendation by coordinating the first-order WPNavNet, and the first-order TermNavNet integrated with the semantic network of Web-pages (TermNetWP). The recommendation strategy (R.WP.AutoTopic.1st.1) is used.

Case 7 (R.WP.AutoTopic.2nd.1) Test the effectiveness of the semantic-enhanced Web-page recommendation by coordinating the second-order WPNavNet, and the second-order TermNavNet integrated with the semantic network of Web-pages (TermNetWP). The recommendation strategy (R.WP.AutoTopic.2nd.1) is used.

Cases 8, *10, 12*, and *14* are similar to Case 6, but the different recommendation strategies are used, as indicated in Table 7-9.

Cases 9, *11*, *13*, and *15* are similar to Case 7, but the different recommendation strategies are used, as indicated in Table 7-9.

Comparisons of Experimental Results

Based on the above experimental cases, seven sets of comparisons are presented. Firstly, the cases using the basic semantically enhanced Web-page recommendation strategies, i.e. Cases 4, 5, 6 and 7, are compared with the Web-page recommendation strategies without semantic enhancement, i.e. Cases 2 and 3, and the base case, i.e. Case 1. Then, the cases using the variations of R.WP.AutoTopic.1st.1 and R.WP.AutoTopic.2nd.1 are compared with Cases 2 and 3, and Case 1. The seven sets of comparisons of experimental cases are summarized Table 7-10, and detailed in the following.

Table 7-10: Comparisons of experimental results

No.	Compared experimental cases
1	4, 5 vs 1, 2, 3
2	4, 5 vs 6, 7
3	6, 7 vs 1, 2, 3
4	8, 9, 12, 13 vs 6, 7
5	10, 11, 14, 15 vs 6, 7
6	6, 8, 10, 12, 14 vs 1
7	7, 9, 11, 13, 15 vs 1

The first set is the comparisons of experimental results of Cases 4,5 against Cases 2,3, using the same model, i.e. WPNavNet, but not semantically enhanced by a constructed domain ontology (DomainOntoWP), to validate the effectiveness of semantics brought in by *the models based on the domain ontology*. Cases 2-5 are also compared with the base Case 1 to validate the effectiveness of *DomainOntoWP* and *WPNavNet* in making Web-page recommendations.

The second set is the comparisons of experimental results of Cases 4,5 against Cases 6,7, respectively, to compare the effectiveness of semantics brought in by the models based on *DomainOntoWP* against the one from the models based on the automatically constructed semantic network of Web-pages (*TermNetWP*).

The third set is the comparisons of the experimental results of Case 6 (R.WP.AutoTopic.1st.1), and Case 7 (R.WP.AutoTopic.2nd.1) against Case 1 (R.PLWAP), Case 2 (R.WP.1st), and Case 3 (R.WP.2nd). This set compares the effectiveness of the *basic*

semantically enhanced Web-page recommendation strategies, i.e. R.WP.AutoTopic.1st.1, and R.WP.AutoTopic.2nd.1, coordinating WPNavNet, and the TermNetWP-based TermNavNet, with the *Web-page recommendation strategies without semantic enhancement*, i.e. R.PLWAP, R.WP.1st, and R.WP.2nd.

The fourth set is the comparisons of the experimental results of Case 8 (R.WP.AutoTopic.1st.2), Case 9 (R.WP.AutoTopic.2nd.2), Case 12 (R.WP.AutoTopic.1st.4), and Case 13 (R.WP.AutoTopic.2nd.4) against Case 6 (R.WP.AutoTopic.1st.1), and Case 7 (R.WP.AutoTopic.2nd.1). This comparison set is used to evaluate the effectiveness of strategies used in Cases 8,9, and 12,13, which can recommend *Web-pages mapped to EACH SET of predicted next domain terms* rather than separate ones, in that each next domain term set is predicted given a current domain term or a pair current and previous domain terms, by basing the automatically constructed semantic knowledge bases, i.e. TermNetWP and TermNavNet. Moreover, next domain term *prediction probabilities* are considered for Web-page recommendation in the strategies in Cases 12,13.

The fifth set is the comparisons of the experimental results of Case 10 (R.WP.AutoTopic.1st.3), Case 11 (R.WP.AutoTopic.2nd.3), Case 14 (R.WP.AutoTopic.1st.5), and Case 15 (R.WP.AutoTopic.2nd.5) against Case 6 (R.WP.AutoTopic.1st.1), and Case 7 (R.WP.AutoTopic.2nd.1). Similarly with the fourth comparison set, this set evaluates the effectiveness of strategies used in Cases 10,11, and 14,15, which can recommend *Web-pages mapped to ALL of predicted next domain terms* by basing the automatically constructed semantic knowledge bases, i.e. TermNetWP and TermNavNet. Moreover, next domain term *prediction probabilities* are considered for Web-page recommendation in the strategies in Cases 14,15.

The sixth set is the comparisons of the experimental results of Case 6 (R.WP.AutoTopic.1st.1), Case 8 (R.WP.AutoTopic.1st.2), Case 10 (R.WP.AutoTopic.1st.3), Case 12 (R.WP.AutoTopic.1st.4), and Case 14 (R.WP.AutoTopic.1st.5) against Case 1 (R.PLWAP). This set compares the effectiveness of strategies used in Cases 6, 8, 10, 12, and 14, which can recommend Web-pages mapped to each predicted next domain term or each set of predicted next domain terms, or all of predicted next domain terms, in that each

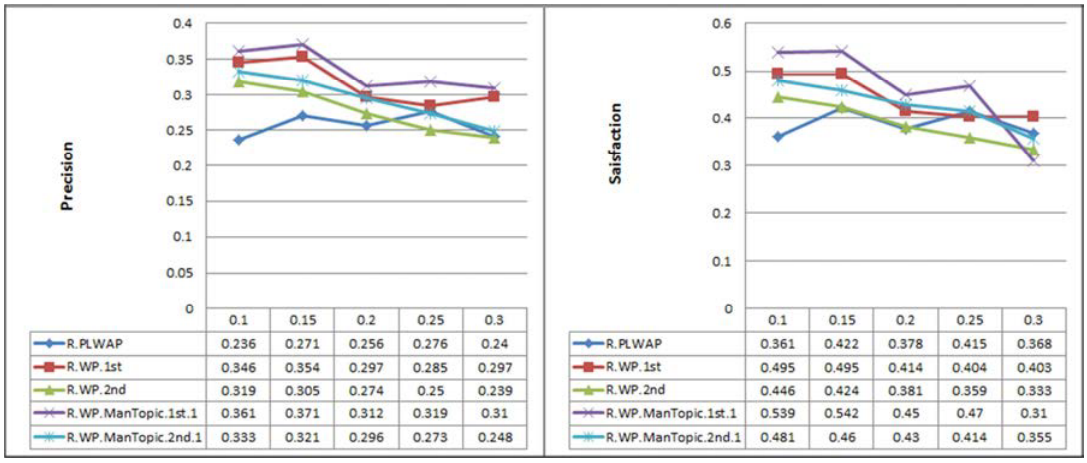
next domain term set is predicted given a current domain term, by basing the automatically constructed semantic knowledge bases, i.e. *TermNetWP* and the *first-order TermNavNet*, with the base case, i.e. R.PLWAP.

The seventh set is the comparisons of the experimental results of Case 7 (R.WP.AutoTopic.2nd.1), Case 9 (R.WP.AutoTopic.2nd.2), Case 11 (R.WP.AutoTopic.2nd.3), Case 13 (R.WP.AutoTopic.2nd.4), and Case 15 (R.WP.AutoTopic.2nd.5) against Case 1 (R.PLWAP). This set compares the effectiveness of strategies used in Cases 7, 9, 11, 13, and 15, which can recommend Web-pages mapped to each predicted next domain term or each set of predicted next domain terms, or all of predicted next domain terms, in that each next domain term set is predicted given a pair current and previous domain terms, by basing the automatically constructed semantic knowledge bases, i.e. *TermNetWP* and the *second-order TermNavNet*, with the base case, i.e. R.PLWAP.

It is noted that, because DomainOntoWP is constructed for 173 Web-pages in the domain of Microsoft software products, these Web-pages only are involved in the dataset for the experimental cases in the comparison sets using DomainOntoWP.

1) Comparison of experimental results for Cases 1-5

The experimental Cases 1-5 were run with different chosen MinSup values, which are 0.1%, 0.15%, 0.2%, 0.25%, and 0.3%. The evaluation measure values (Precision and Satisfaction) with these MinSup values for the five experimental cases are depicted in Figure 7-2, where the horizontal axis represents MinSup, the vertical axis on the left part represents Precision, and the one on the right part represents Satisfaction. The used strategies are shown in regard to the experimental cases, respectively, in Figure 7-2.



	0.1	0.15	0.2	0.25	0.3
R.PLWAP	0.236	0.271	0.256	0.276	0.24
R.WP.1st	0.346	0.354	0.297	0.285	0.297
R.WP.2nd	0.319	0.305	0.274	0.25	0.239
R.WP.ManTopic.1st.1	0.361	0.371	0.312	0.319	0.31
R.WP.ManTopic.2nd.1	0.333	0.321	0.296	0.273	0.248

	0.1	0.15	0.2	0.25	0.3
R.PLWAP	0.361	0.422	0.378	0.415	0.368
R.WP.1st	0.495	0.495	0.414	0.404	0.403
R.WP.2nd	0.446	0.424	0.381	0.359	0.333
R.WP.ManTopic.1st.1	0.539	0.542	0.45	0.47	0.31
R.WP.ManTopic.2nd.1	0.481	0.46	0.43	0.414	0.355

Figure 7-2: Results for experimental Cases 1-5

From Figure 7-2, the following observations are made:

- Both Cases 2 (R.WP.1st) and 4 (R.WP.ManTopic.1st.1) significantly outperform Case 1 (R.PLWAP) in terms of both measures for almost all the MinSup values. Especially, the precision of R.WP.ManTopic.1st.1 is always highest and peaks at the MinSup of 0.15%.
- Both Cases 3 (R.WP.2nd) and 5 (R.WP.ManTopic.2nd.1) don't perform so well as Cases 2 and 4 in terms of both measures, but their precisions are higher than the precision of Case 1 for the MinSup values of {0.1%, 0.15%, 0.2%}. Moreover, the performance of R.WP.ManTopic.2nd.1 is always higher than that of R.WP.2nd.

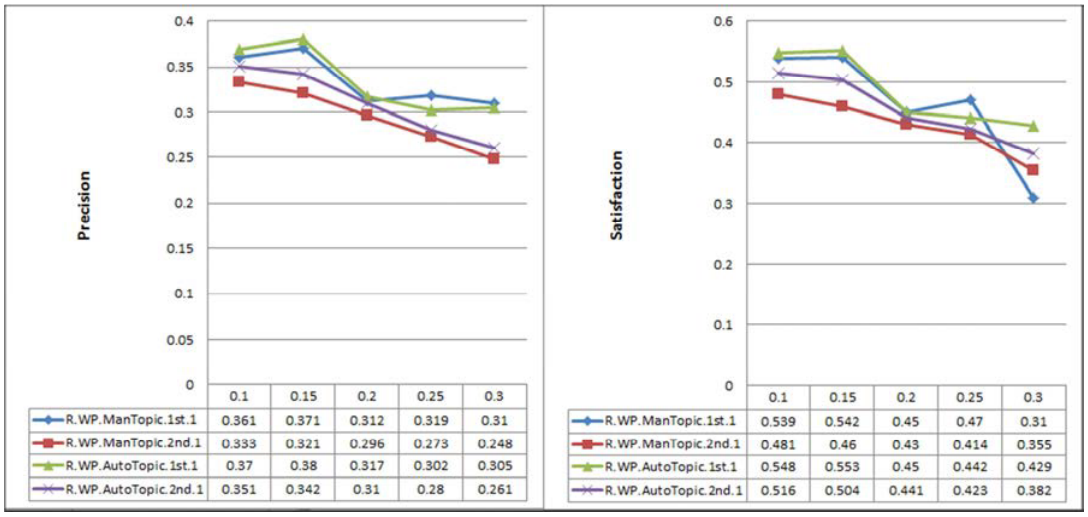
- Using the same WPNavNet model, the first-order navigation model used in R.WP.1st is better than the second-order navigation model used in R.WP.2nd.

An important thing is that the highest performance of each strategy using the proposed models in Cases 2-5 is higher than the one using PLWAP-Mine in Case 1. Case 4 (R.WP.ManTopic.1st.1) can gain the highest performance in overall. This indicates that the WPNavNet model is effective for Web-page recommendation, and adding semantic information into the Web-page recommendation model can enhance the recommendation results significantly. That is because the DomainOntoWP-based TermNavNet model can overcome the "new page" problem, i.e. some pages in the testing WAS are not in the

discovered Web-page patterns, which the WPNavNet model and PLWAP-Mine fail to solve.

2) Comparison of experimental results for Cases 4-7

The experimental Cases 4-7 were run at different chosen MinSup which are 0.1%, 0.15%, 0.2%, 0.25%, and 0.3%. The evaluation measure values (Precision and Satisfaction) with these MinSup values for the four experimental cases are depicted in Figure 7-3, where the horizontal axis represents the MinSup, the vertical axis on the left part represents Precision, and the one on the right part represents Satisfaction. The used strategies are shown in regard to the experimental cases, respectively, in Figure 7-3.



	0.1	0.15	0.2	0.25	0.3
R.WP.ManTopic.1st.1	0.361	0.371	0.312	0.319	0.31
R.WP.ManTopic.2nd.1	0.333	0.321	0.296	0.273	0.248
R.WP.AutoTopic.1st.1	0.37	0.38	0.317	0.302	0.305
R.WP.AutoTopic.2nd.1	0.351	0.342	0.31	0.28	0.261

	0.1	0.15	0.2	0.25	0.3
R.WP.ManTopic.1st.1	0.539	0.542	0.45	0.47	0.31
R.WP.ManTopic.2nd.1	0.481	0.46	0.43	0.414	0.355
R.WP.AutoTopic.1st.1	0.548	0.553	0.45	0.442	0.429
R.WP.AutoTopic.2nd.1	0.516	0.504	0.441	0.423	0.382

Figure 7-3: Results for experimental Cases 4-7

From Figure 7-3, the following observations are made:

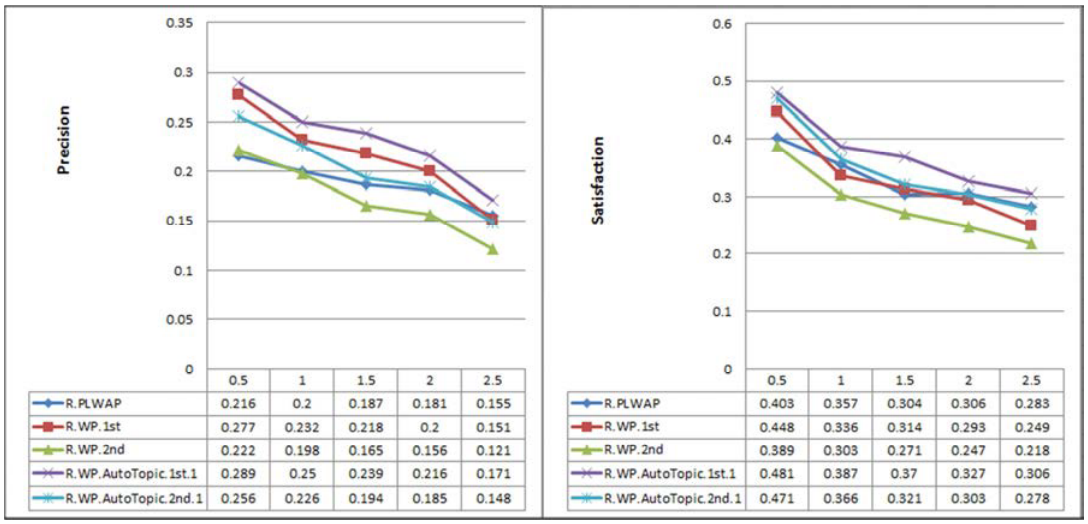
- Cases 4 (R.WP.ManTopic.1st.1) and 6 (R.WP.AutoTopic.1st.1) gain higher precision than Cases 5 (R.WP.ManTopic.2nd.1) and 7 (R.WP.AutoTopic.2nd.1) for all the MinSup values. The performance of R.WP.AutoTopic.1st.1 is slightly higher than that of R.WP.ManTopic.1st.1 in terms of both measures as MinSup = {0.1%, 0.15%, 0.2%}.
- With the same second-order TermNavNet, the performance of TermNetWP-based model in Case 7 is always higher than that of the DomainOntoWP-based model (Case

5).

In general, Case 6 using R.WP.AutoTopic.1st.1 can gain the highest performance, compared with the remaining cases. The results indicate that the first-order navigation models are more effective than the second-order navigation models in both types of strategies based on DomainOntoWP and TermNetWP. The TermNetWP-based models better raises the performance of Web-page recommendation than the DomainOntoWP-based models.

3) Comparison of experimental results for Cases 1, 2, 3, 6, and 7

The experimental Cases 1, 2, 3, 6, and 7 were run at different MinSup which are 0.5%, 1%, 1.5%, 2%, and 2.5%. The evaluation measure values (Precision and Satisfaction) with these MinSup values for the five experimental cases are depicted in Figure 7-4, where the horizontal axis represents the MinSup, the vertical axis on the left part represents Precision, and the one on the right part represents Satisfaction. The used strategies are shown in regard to the experimental cases, respectively, in Figure 7-4.



	0.5	1	1.5	2	2.5
R.PLWAP	0.216	0.2	0.187	0.181	0.155
R.WP.1st	0.277	0.232	0.218	0.2	0.151
R.WP.2nd	0.222	0.198	0.165	0.156	0.121
R.WP.AutoTopic.1st.1	0.289	0.25	0.239	0.216	0.171
R.WP.AutoTopic.2nd.1	0.256	0.226	0.194	0.185	0.148

	0.5	1	1.5	2	2.5
R.PLWAP	0.403	0.357	0.304	0.306	0.283
R.WP.1st	0.448	0.336	0.314	0.293	0.249
R.WP.2nd	0.389	0.303	0.271	0.247	0.218
R.WP.AutoTopic.1st.1	0.481	0.387	0.37	0.327	0.306
R.WP.AutoTopic.2nd.1	0.471	0.366	0.321	0.303	0.278

Figure 7-4: Results for experimental Cases 1, 2, 3, 6, and 7

From Figure 7-4, the following observations are made:

- There are trends of decrease in the performance of the experimental cases in this

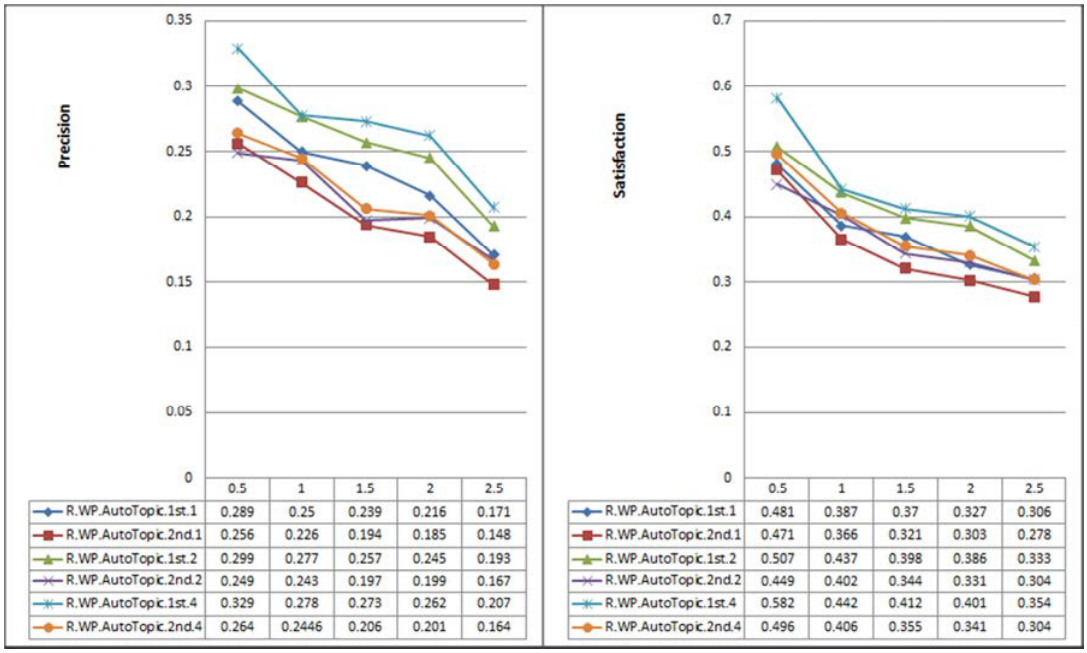
comparison set when the MinSup increases. This reasonably reflects the fact that the all recommendation strategies rely on FWAP. Increase in MinSup will lead to decrease in the size of FWAP, that is decrease in the quality of the recommendation strategies. However, the important thing is which strategy can gain the highest performance for the chosen MinSup values.

- The performance of strategy R.WP.AutoTopic.1st.1 in Case 6 is higher than that of the other strategies for all the MinSup values.
- The satisfaction of strategy R.WP.AutoTopic.2nd.1 in Case 7 is higher than that of strategies R.PLWAP, R.WP.1st, and R.WP.2nd for the MinSup values of 0.5%, 1%, and 1.5%.
- Similar to the first comparison set, the precision of R.WP.1st in Case 2 is higher than that of R.PLWAP in Case 1 for almost all the MinSup values, as MinSup = {0.5%, 1%, 1.5%, 2%}. While, Case 3 using R.WP.2nd is lower than the others for almost all the MinSup values in terms of both measures.

In overall, the semantic-enhanced models perform Web-page recommendation more effectively than PLWAP-Mine as well as WPNavNet. In these models, the semantic network-based first-order domain term navigation model (Case 6) is the best model to Web-page recommendation since it is coordinated with the first-order Web-page navigation model (WPNavNet).

4) Comparison of experimental results for Cases 6-9 and 12,13

The experimental Cases 6-9 and 12-13 were run with different chosen MinSup values, which are 0.5%, 1%, 1.5%, 2%, and 2.5%. The evaluation measure values (Precision and Satisfaction) with these MinSup values for the six experimental cases are depicted in Figure 7-5, where the horizontal axis represents the MinSup, the vertical axis on the left part represents Precision, and the one on the right part represents Satisfaction. The used strategies are shown in regard to the experimental cases, respectively.



	0.5	1	1.5	2	2.5
R.WP.AutoTopic.1st.1	0.289	0.25	0.239	0.216	0.171
R.WP.AutoTopic.2nd.1	0.256	0.226	0.194	0.185	0.148
R.WP.AutoTopic.1st.2	0.299	0.277	0.257	0.245	0.193
R.WP.AutoTopic.2nd.2	0.249	0.243	0.197	0.199	0.167
R.WP.AutoTopic.1st.4	0.329	0.278	0.273	0.262	0.207
R.WP.AutoTopic.2nd.4	0.264	0.2446	0.206	0.201	0.164

	0.5	1	1.5	2	2.5
R.WP.AutoTopic.1st.1	0.481	0.387	0.37	0.327	0.306
R.WP.AutoTopic.2nd.1	0.471	0.366	0.321	0.303	0.278
R.WP.AutoTopic.1st.2	0.507	0.437	0.398	0.386	0.333
R.WP.AutoTopic.2nd.2	0.449	0.402	0.344	0.331	0.304
R.WP.AutoTopic.1st.4	0.582	0.442	0.412	0.401	0.354
R.WP.AutoTopic.2nd.4	0.496	0.406	0.355	0.341	0.304

Figure 7-5: Results for experimental Cases 6-9 and 12,13

From Figure 7-5, the following observations are made:

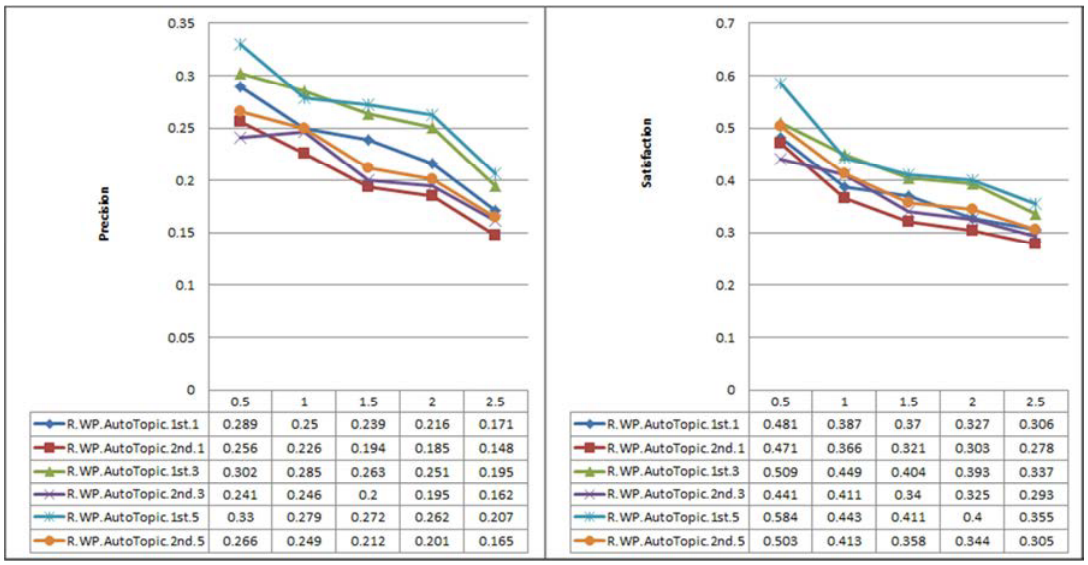
- Both Cases 8 (R.WP.AutoTopic.1st.2) and 12 (R.WP.AutoTopic.1st.4) significantly outperform the other cases in terms of both measures for all the MinSup values. In particular, R.WP.AutoTopic.1st.4 in Case 12 can gain the highest performance, and R.WP.AutoTopic.1st.2 in Case 8 can gain the second highest one.
- Based on the same second-order domain term navigation model, strategies R.WP.AutoTopic.2nd.2 and R.WP.AutoTopic.2nd.4 in Cases 9 and 13, respectively, perform better than strategy R.WP.AutoTopic.2nd.1 in Case 7 for almost all the MinSup values, as MinSup = {1%, 1.5%, 2%, 2.5%}.

In summary, by the navigation models with the same order, e.g. first-order or second-order, recommending Web-pages mapped to each set of predicted next domain terms (Cases 8,9, and 12,13) is more effective than recommending Web-pages mapped to each of the predicted next domain terms (Cases 6,7). Based on the same TermNetWP model, the first-order TermNavNet model (in Cases 6, 8, and 12) is more effective than the second-

order TermNavNet model (in Cases 7, 9, and 13) in Web-page recommendation. R.WP.AutoTopic.1st.4 is similar to R.WP.AutoTopic.1st.2, but has next domain term prediction probabilities taken into account, hence R.WP.AutoTopic.1st.4 in Case 12 can make better Web-page recommendation results.

5) Comparison of experimental results for Cases 6, 7, 10, 11, 14, and 15

The experimental Cases 6, 7, 10, 11, 14, and 15 were run with different chosen MinSup values, which are 0.5%, 1%, 1.5%, 2%, and 2.5%. The evaluation measure values (Precision and Satisfaction) with these MinSup values for the six experimental cases are depicted in Figure 7-6, where the horizontal axis represents the MinSup, the vertical axis on the left part represents Precision, and the one on the right part represents Satisfaction. The used strategies are shown in regard to the experimental cases, respectively, in Figure 7-6.



	0.5	1	1.5	2	2.5
R.WP.AutoTopic.1st.1	0.289	0.25	0.239	0.216	0.171
R.WP.AutoTopic.2nd.1	0.256	0.226	0.194	0.185	0.148
R.WP.AutoTopic.1st.3	0.302	0.285	0.263	0.251	0.195
R.WP.AutoTopic.2nd.3	0.241	0.246	0.2	0.195	0.162
R.WP.AutoTopic.1st.5	0.33	0.279	0.272	0.262	0.207
R.WP.AutoTopic.2nd.5	0.266	0.249	0.212	0.201	0.165

	0.5	1	1.5	2	2.5
R.WP.AutoTopic.1st.1	0.481	0.387	0.37	0.327	0.306
R.WP.AutoTopic.2nd.1	0.471	0.366	0.321	0.303	0.278
R.WP.AutoTopic.1st.3	0.509	0.449	0.404	0.393	0.337
R.WP.AutoTopic.2nd.3	0.441	0.411	0.34	0.325	0.293
R.WP.AutoTopic.1st.5	0.584	0.443	0.411	0.4	0.355
R.WP.AutoTopic.2nd.5	0.503	0.413	0.358	0.344	0.305

Figure 7-6: Results for experimental Cases 6, 7, 10, 11, 14, and 15

From Figure 7-6, the following observations are made:

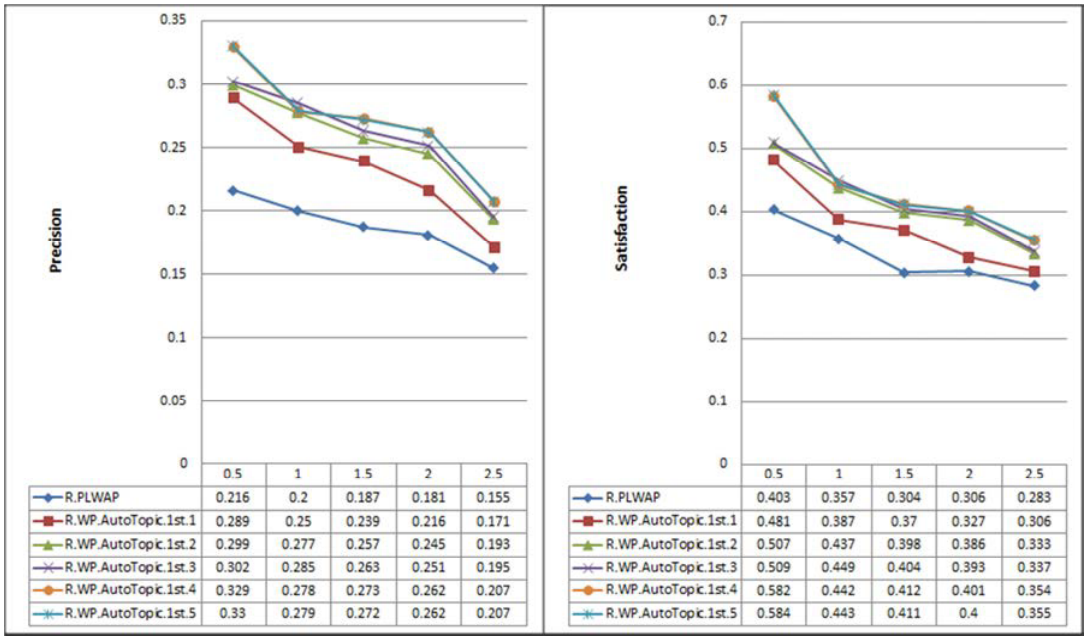
- Strategies R.WP.AutoTopic.1st.3 in Case 10, and R.WP.AutoTopic.1st.5 in Case 14 outperform the other strategies in terms of both measures for all the MinSup values. In particular, R.WP.AutoTopic.1st.5 in Case 14 can gain the highest performance, and R.WP.AutoTopic.1st.3 in Case 10 can gain the second highest one.

- Based on the same second-order domain term navigation model, strategies R.WP.AutoTopic.2nd.3 and R.WP.AutoTopic.2nd.5 in Cases 11 and 15, respectively, perform better than strategy R.WP.AutoTopic.2nd.1 in Case 7 for almost all the MinSup values, as MinSup = {1%, 1.5%, 2%, 2.5%}.

Generally, by the navigation models with the same order, e.g. first-order or second-order, recommending Web-pages mapped to all of predicted next domain terms (Cases 10,11, and 14,15) is more effective than recommending Web-pages mapped to each of the predicted next domain terms (Cases 6,7). Based on the same TermNetWP model, the first-order TermNavNet model (in Cases 6, 10, and 14) is more effective than the second-order TermNavNet model (in Cases 7, 11, and 15) in recommendation. Moreover, R.WP.AutoTopic.1st.5 is similar to R.WP.AutoTopic.1st.3, but has next domain term prediction probabilities taken into account, hence R.WP.AutoTopic.1st.5 in Case 14 can make better Web-page recommendation results.

6) Comparison of experimental results for Cases 1, 6, 8, 10, 12, and 14

The experimental Cases 1, 6, 8, 10, 12, and 14 were run with different chosen MinSup values, which are 0.5%, 1%, 1.5%, 2%, and 2.5%. The evaluation measure values (Precision and Satisfaction) with these MinSup values for the six experimental cases are depicted in Figure 7-7, where the horizontal axis represents the MinSup, the vertical axis on the left part represents Precision, and the one on the right part represents Satisfaction. The used strategies are shown in regard to the experimental cases, respectively, in Figure 7-7.



	0.5	1	1.5	2	2.5
R.PLWAP	0.216	0.2	0.187	0.181	0.155
R.WP.AutoTopic.1st.1	0.289	0.25	0.239	0.216	0.171
R.WP.AutoTopic.1st.2	0.299	0.277	0.257	0.245	0.193
R.WP.AutoTopic.1st.3	0.302	0.285	0.263	0.251	0.195
R.WP.AutoTopic.1st.4	0.329	0.278	0.273	0.262	0.207
R.WP.AutoTopic.1st.5	0.33	0.279	0.272	0.262	0.207

	0.5	1	1.5	2	2.5
R.PLWAP	0.403	0.357	0.304	0.306	0.283
R.WP.AutoTopic.1st.1	0.481	0.387	0.37	0.327	0.306
R.WP.AutoTopic.1st.2	0.507	0.437	0.398	0.386	0.333
R.WP.AutoTopic.1st.3	0.509	0.449	0.404	0.393	0.337
R.WP.AutoTopic.1st.4	0.582	0.442	0.412	0.401	0.354
R.WP.AutoTopic.1st.5	0.584	0.443	0.411	0.4	0.355

Figure 7-7: Results for experimental Cases 1, 6, 8, 10, 12, and 14

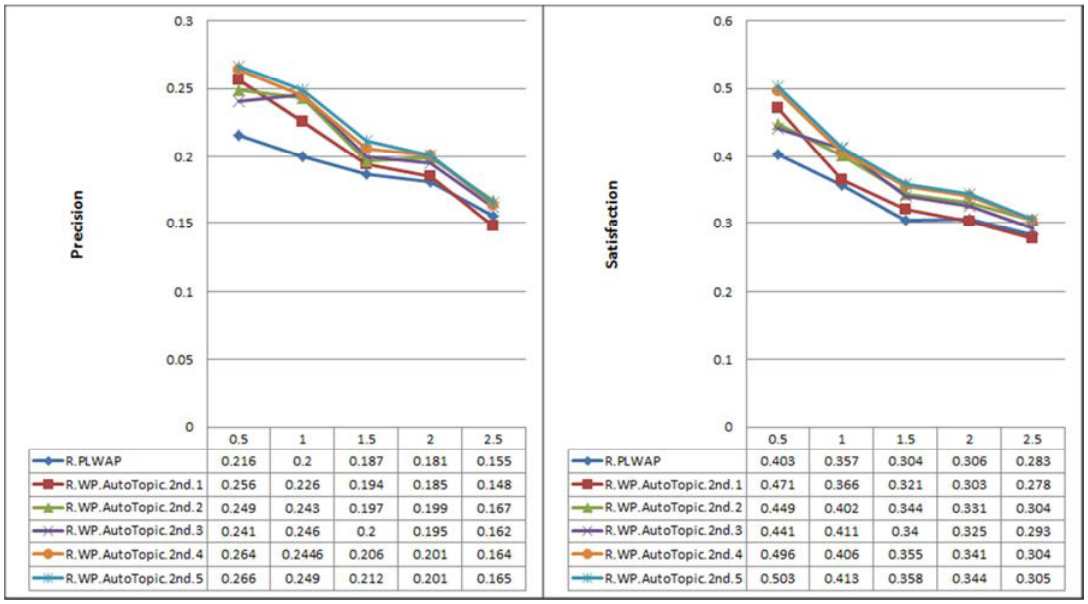
From Figure 7-7, the following observations are made:

- Cases 12 (R.WP.AutoTopic.1st.4) and 14 (R.WP.AutoTopic.1st.5) outperform the others in terms of both measures for almost all the MinSup values. Especially, their performance is significantly higher than Cases 1 (R.PLWAP) and 6 (R.WP.AutoTopic.1st.1).
- The performance of Cases 8 and 12 is similar with the one of Cases 10 and 14, respectively, for all the MinSup values. This shows that making Web-page recommendations based on all sets of predicted next domain terms or each set of ones, in which each set of next domain terms is predicted given each current domain term, achieves similar results, by using the first-order navigation models.

It clearly points out that strategies R.WP.AutoTopic.1st.4 in Case 12 and R.WP.AutoTopic.1st.5 in Case 14, which involve prediction probabilities assigned to next domain terms in Web-page recommendation using TermNetWP and the first-order TermNavNet, can gain the best performance in this comparison set.

7) Comparison of experimental results for Cases 1, 7, 9, 11, 13, and 15

In this comparison set, we run the experimental Cases 1, 7, 9, 11, 13, and 15 at different chosen MinSup values, which are 0.5%, 1%, 1.5%, 2%, and 2.5%. The evaluation measure values (Precision and Satisfaction) with these MinSup values for the six experimental cases are depicted in Figure 7-8, where the horizontal axis represents MinSup, the vertical axis on the left part represents Precision, and the one on the right part represents Satisfaction. The used strategies are shown in regard to the experimental cases, respectively, in Figure 7-8.



	0.5	1	1.5	2	2.5
R.PLWAP	0.216	0.2	0.187	0.181	0.155
R.WP.AutoTopic.2nd.1	0.256	0.226	0.194	0.185	0.148
R.WP.AutoTopic.2nd.2	0.249	0.243	0.197	0.199	0.167
R.WP.AutoTopic.2nd.3	0.241	0.246	0.2	0.195	0.162
R.WP.AutoTopic.2nd.4	0.264	0.2446	0.206	0.201	0.164
R.WP.AutoTopic.2nd.5	0.266	0.249	0.212	0.201	0.165

	0.5	1	1.5	2	2.5
R.PLWAP	0.403	0.357	0.304	0.306	0.283
R.WP.AutoTopic.2nd.1	0.471	0.366	0.321	0.303	0.278
R.WP.AutoTopic.2nd.2	0.449	0.402	0.344	0.331	0.304
R.WP.AutoTopic.2nd.3	0.441	0.411	0.34	0.325	0.293
R.WP.AutoTopic.2nd.4	0.496	0.406	0.355	0.341	0.304
R.WP.AutoTopic.2nd.5	0.503	0.413	0.358	0.344	0.305

Figure 7-8: Results for experimental Cases 1, 7, 9, 11, 13, and 15

From Figure 7-8, the following observations are made:

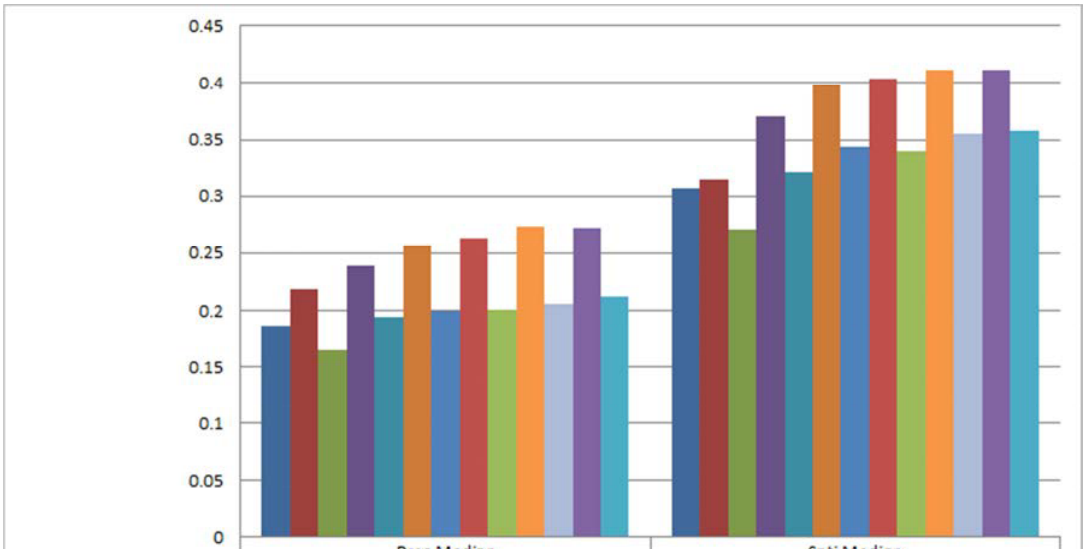
- Cases 13 (R.WP.AutoTopic.2nd.4) and 15 (R.WP.AutoTopic.2nd.5) outperform the others in terms of both measures for almost all the MinSup values. Especially, their performance is significantly higher than Cases 1 (R.PLWAP) and 7 (R.WP.AutoTopic.2nd.1).
- The performance of Cases 9 and 13 is similar with the one of Cases 11 and 15, respectively, for all the MinSup values. This shows that making Web-page recommendations based on all sets of predicted next domain terms or each set of

ones, in which each set of next domain terms is predicted given each pairs of current and previous domain terms, achieves similar results, by using the second-order navigation models.

It clearly points out that strategies R.WP.AutoTopic.2nd.4 in Case 13 and R.WP.AutoTopic.2nd.5 in Case 15, which involve prediction probabilities assigned to next domain terms in Web-page recommendation using TermNetWP and the second-order TermNavNet, can gain the best performance in this comparison set.

Remarks

From the seven comparison sets of the experimental cases, it has been found that the proposed domain knowledge representation models, i.e. DomainOntoWP and TermNetWP, are effective to semantically enrich and enhance Web-page recommendation. The TermNetWP-based models are more effective than the DomainOntoWP-based models in integration with the domain term navigation model for supporting Web-page recommendation. Moreover, the TermNetWP-based models do not depend on the human knowledge in the domain knowledge construction like the DomainOntoWP-based models. This shows that the models of automatic knowledge construction are more promising in Web-page recommendation. Therefore, this part remarks the cases using the automatically constructed knowledge representation models, i.e. WPNavNet, TermNetWP, and TermNavNet, in a general view. Figure 7-9 depicts the medians of *precisions* and *satisfactions* of Cases 1-3, and 6-15 which have been carried out in the comparison sets 3-7.



	Prec.Median	Sati.Median
R.PLWAP	0.187	0.306
R.WP.1st	0.218	0.314
R.WP.2nd	0.165	0.271
R.WP.AutoTopic.1st.1	0.239	0.37
R.WP.AutoTopic.2nd.1	0.194	0.321
R.WP.AutoTopic.1st.2	0.257	0.398
R.WP.AutoTopic.2nd.2	0.199	0.344
R.WP.AutoTopic.1st.3	0.263	0.404
R.WP.AutoTopic.2nd.3	0.2	0.34
R.WP.AutoTopic.1st.4	0.273	0.412
R.WP.AutoTopic.2nd.4	0.206	0.355
R.WP.AutoTopic.1st.5	0.272	0.411
R.WP.AutoTopic.2nd.5	0.212	0.358

Figure 7-9: The medians of Precisions and Satisfactions of Cases 1-3, 6-15

As shown in Figure 7-9, R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 can achieve the highest positions, followed by R.WP.AutoTopic.1st.3, R.WP.AutoTopic.1st.2, and R.WP.AutoTopic.1st.1. It is explained that making Web-page recommendations based on all sets of next domain terms (R.WP.AutoTopic.1st.5) or each set of next domain terms (R.WP.AutoTopic.1st.4), in which each set of next domain terms are predicted given a current domain term by using the TermNetWP-based first-order TermNavNet models, can well support the first-order Web-page navigation model in enhancing the pool of recommended Web-pages. Involving probabilities into next domain term prediction can boost the making Web-page recommendations to achieve the highest performance.

However, the TermNetWP-based second-order TermNavNet model is not so efficient as the TermNetWP-based first-order TermNavNet model in the all experimental cases. In other words, second-order probabilities might not suitable for the navigation models for Web-page prediction. This seems surprising, but the reason is the number of the predicted next states by the second-order navigation models is limited by second-order probabilities, and the system cannot choose the best rules for prediction. While, the first-order navigation models do not look far into the past to discriminate correctly the different states of the generative process, so there are more choices for prediction. Moreover, the first-order navigation models are less complicated than the second-order navigation models in terms of state-space and probability computation. Therefore, the first-order navigation models are preferred in the Web-page recommendation process.

In overall, the experimental results show that the semantically enhanced Web-page recommendation strategies can achieve higher performance than the Web-page recommendation strategies without semantic enhancement. In other words, the coordiantion between the Web-page navigation model, and the domain ontology-based domain term navigation model or the semantic network-based domain term navigation model can significantly improve Web-page recommendation. This validates the proposed knowledge representation models, i.e. DomainOntoWP, TermNetWP, WPNavNet, and TermNavNet, which are achievable and promising in semantic-enhanced Web-page recommender systems.

7.4.4. Experiments with Real World Dataset

As experimented in Sub-section 7.4.3, Web-page recommendation strategies R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 are the best, but how well they perform on a real world dataset. Therefore, this sub-section elaborates on the performance evaluation of the two strategies by using the UTS:HB website (*handbook.uts.edu.au*) in the experiments of Web-page recommendation. This sub-section firstly concerns data preparation for the experiments, and then evaluates experimental results.

Data Preparation

A Web log file of website *handbook.uts.edu.au* was collected in a period of time from April 19th, 2011 to May 25th, 2011. The Web log is quite large with 11456 accessed Web-pages. For experimenting, 700 firstly visited Web-pages are selected because they are most frequently visited. It is necessary to collect the datasets of Web usage which includes a set of visited Web-page sequences from the Web log. It is very important to select a good dataset from the Web log to acquire knowledge useful for Web-page recommendation. A dataset is good if it does not contain many sequences of length 1. Therefore, a process of selecting datasets appropriate for the Web-page recommendation process was carried out as follows:

- Dataset for each day was created and observed. It has been found that the number of accessed Web-pages is most from 13:00 to 20:00 every day. So, datasets for this period of time from 19/04/2011 to 25/05/2011 were collected. It was observed that the datasets of dates 25th-30th of April, 20th, 22nd, and 25th of May are good as the number of Web access sequences of length 1 is not too many.
- For each of those datasets, 20% of the data for the testing data was selected and the remaining 80% was left for training. According the performance evaluation algorithm (Algorithm 3-5), PLWAP-Mine is used to preliminarily test Web-page recommendation performance for the given datasets. The MinSup values are chosen from 0.3% to 3% for testing. It has been found that the datasets of dates 27th and 28th of April are good for training as we can achieve the average precision higher than the testing results using the other datasets.
- Furthermore, when the datasets of dates 27th and 28th of April are combined as a dataset for training, and each second half of the datasets of dates 30th of April, 20th and 22nd of May is in turn used for testing, the resulting satisfactions are acceptable to be about 0.1. This verifies that the training data is an appropriate sample for the experiments.

Consequently, the combined dataset of dates 27th and 28th of April, referred to as *dsUtshb27-28Apr*, is used as the training data in the system. The testing sets are:

- The second half of the dataset of date 30th of April, named *dsUtshb30Apr*,
- The second half of the dataset of date 20th of May, named *dsUtshb20May*, and
- The second half of the dataset of date 22nd of May, named *dsUtshb22May*.

After obtaining the appropriate datasets, processing data and conducting knowledge bases have been carried out. The models of automatic knowledge construction are applied, which are TermNetWP and the first-order TermNavNet, in the system. The following evaluates experimental results given the selected datasets using the two Web-page recommendation strategies, namely R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5, against the base strategy, namely R.PLWAP, using PLWAP-Mine.

Evaluation and Remarks

As mentioned in Section 7.4.1, and Section 7.4.2, Web-page recommendation performance is evaluated based on the measures of *precision* and *satisfaction*. In a similar way as the experiments in Sub-section 7.4.3, Algorithm 7-8 in Sub-section 7.4.2 is used for performance evaluation. Three experiments are performed at the differently chosen MinSup values, which are 0.3%, 0.4%, 0.5%, 0.6%, 0.7%, 0.8%, and 0.9%, and the recommendation length is set to 10. The training data is *dsUtshb27-28Apr*. The testing sets, i.e. *dsUtshb30Apr*, *dsUtshb20May*, *dsUtshb22May*, are respectively used in the three experiments in order to validate and to compare the performance of the Web-page recommendation model using the proposed strategies, e.g. R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5, with the performance of the Web-page recommendation model using the base strategy (R.PLWAP). Because the size of *dsUtshb27-28Apr* is rather small to obtain high precisions, the satisfaction is used to measure the performance of the recommendation strategies in the three experiments. The trends of precision and satisfaction are often similar, but satisfaction values are often higher than precision values. Hence, we can compare the performance of recommendation strategies using the satisfaction measure.

1) Experiment 1: dsUtshb30Apr is used for testing

The experimental results in experiment 1 are shown in Figure 7-10, which illustrates the

satisfactions of strategies R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 against R.PLWAP.

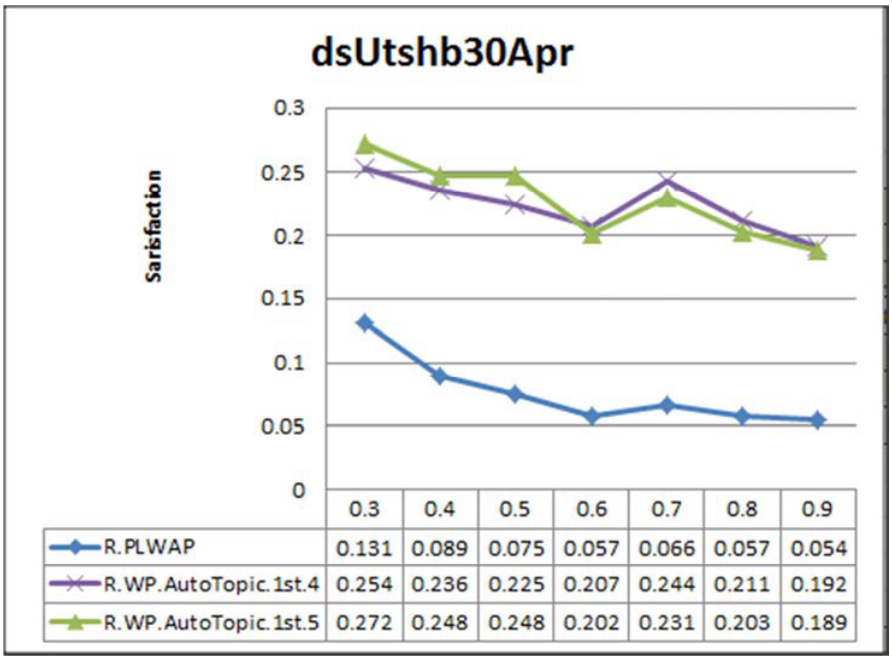
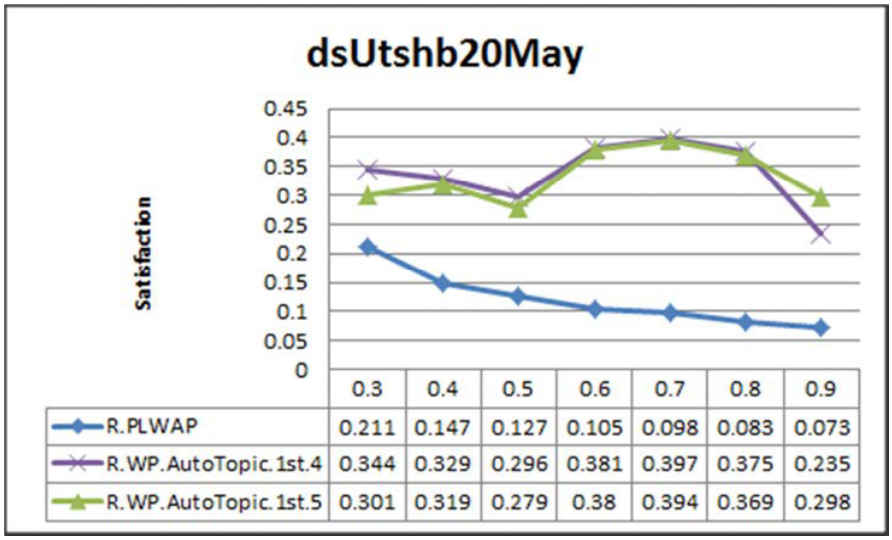


Figure 7-10: Experiment 1: the resulting satisfactions of R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 vs R.PLWAP

From Figure 7-10, strategies R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5, which use the TermNetWP-based first-order TermNavNet model, outperform the base strategy using PLWAP-Mine for all the chosen MinSup values. The satisfactions of R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 are slightly similar and peak at the MinSup of 0.3%.

2) Experiment 2: dsUtshb20May is used for testing

The experimental results in experiment 2 are shown in Figure 7-11, which illustrates the satisfactions of strategies R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 against R.PLWAP.



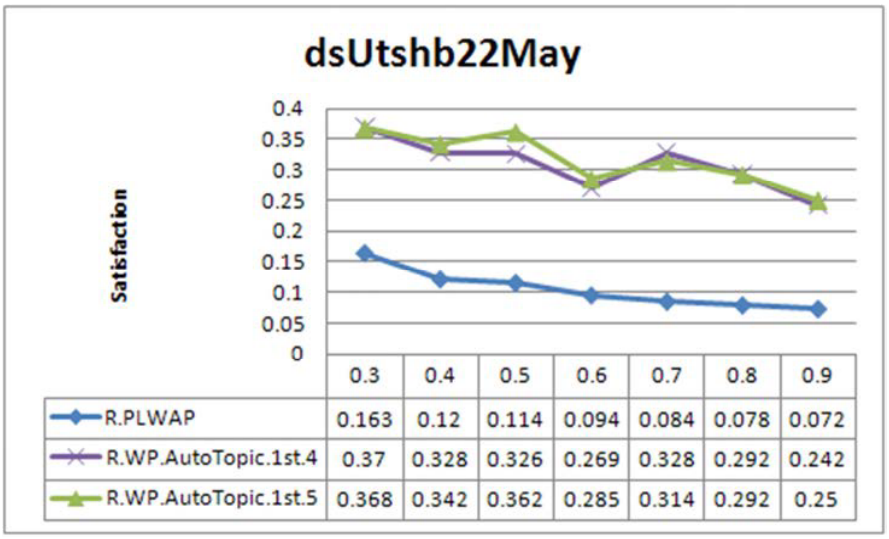
	0.3	0.4	0.5	0.6	0.7	0.8	0.9
R.PLWAP	0.211	0.147	0.127	0.105	0.098	0.083	0.073
R.WP.AutoTopic.1st.4	0.344	0.329	0.296	0.381	0.397	0.375	0.235
R.WP.AutoTopic.1st.5	0.301	0.319	0.279	0.38	0.394	0.369	0.298

Figure 7-11: Experiment 2: the resulting satisfactions of R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 vs R.PLWAP

From Figures 7-11, strategies R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 outperform R.PLWAP for all the chosen MinSup values. However, while there is a trend of decrease in the satisfaction of R.PLWAP when the MinSup increases, the satisfactions of R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 increase when the MinSup increases from 0.5% to 0.7%, and reach the highest point about 0.39.

3) Experiment 3: dsUtshb22May is used for testing

The experimental results in experiment 3 are shown in Figure 7-12, which illustrates the satisfactions of strategies R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 against R.PLWAP.



	0.3	0.4	0.5	0.6	0.7	0.8	0.9
R.PLWAP	0.163	0.12	0.114	0.094	0.084	0.078	0.072
R.WP.AutoTopic.1st.4	0.37	0.328	0.326	0.269	0.328	0.292	0.242
R.WP.AutoTopic.1st.5	0.368	0.342	0.362	0.285	0.314	0.292	0.25

Figure 7-12: Experiment 3: the resulting satisfactions of R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 vs R.PLWAP

From Figure 7-12, strategies R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 still outperform PLWAP-Mine for all the chosen MinSup values. Trends in the performance of R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 are similar to the performance trends of R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 in Experiment 1. The satisfactions of them gain approximate 0.37 at the MinSup of 0.3%.

From the three experiments, the trends in the performance in Experiments 1 and 3 are similar, but different from the performance trends in Experiment 2. Observing the testing datasets in the three experiments, it has been found that *dsUtshb20May* in Experiment 2 differs from the other datasets in terms of the length of Web access sequences in the datasets. In *dsUtshb20May*, there are few sequences of length longer than 80, and the number of sequences of length 1 is more than those in the other testing datasets. In other words, the lengths of sequences in *dsUtshb20May* are quite very unequal. The lengths of sequences in *dsUtshb30Apr* and *dsUtshb22May* are rather equal. Consequently, the obtained performance of strategies R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 in Experiment 2 fluctuates more widely than the one in Experiments 1 and 3.

In overall, the Web-page recommendation model using the proposed knowledge representation models, i.e., WPNavNet and the TermNetWP-based first-order

TermNavNet, are more efficient to make Web-page recommendations than using solely PLWAP-Mine. The performance of R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 are similar in the all experiments, so we can choose one of them for making Web-page recommendations at a certain website.

7.5. Discussions

Regarding the Web-page recommendation performance of the knowledge representation models of Web-pages, the TermNetWP models are mostly more effective than the DomainOntoWP models. This is because the automatically constructed semantic network of Web-pages, i.e. TermNetWP, has the following advantages:

- TermNetWP allows sorting of domain terms and Web-pages in the reasoning algorithms, so the most possible items are taken into account in the domain term prediction and Web-page recommendation processes.

- TermNetWP quite fully represents domain terms of Web-pages, and the relationships between domain terms and Web-pages, so the meaning of almost all Web-pages can be understood and processed by the machine.

The manually constructed domain ontology of a website, i.e. DomainOntoWP, can enhance the semantic information of Web-pages, but the manual construction is subjective, and is therefore limited by the developer's knowledge. Whilst the manual approach cannot be reused in other websites, the automatic approach can be applied to any websites. By using the automatic approach, the system can adopt a particular Web usage data and automatically construct semantic knowledge bases for Web-page recommendation, as deployed in the cases of the MS and UTS:HB websites.

7.6. Summary

This chapter has proposed the new framework of SWRS coordinating the domain knowledge representation models, i.e. DomainOntoWP and TermNetWP, the transformed Web usage knowledge representation model, i.e. WPNavNet, and the semantic Web usage knowledge representation model, i.e. TermNavNet, for efficiently making Web-page recommendations. Fourteen various recommendation strategies have been proposed and classified into three types: (1) without semantic enhancement using WPNavNet, (2) with semantic enhancement based on the domain ontology by coordinating WPNavNet and the DomainOntoWP-based TermNavNet, (3) with semantic enhancement based on the semantic network by coordinating WPNavNet and the TermNetWP-based TermNavNet.

The thorough evaluation using the public and real world datasets has shown that the semantically enhanced recommendation strategies outperform the PLWAP-Mine based strategy. Especially, the recommendation strategies using the first-order navigation model in the three types are outstanding and significantly outperform the PLWAP-Mine based strategy. This validates the effectiveness of the proposed knowledge representation models in the SWRS.

Chapter 8.

A SEMANTIC-ENHANCED WEB-PAGE RECOMMENDER SYSTEM PROTOTYPE

8.1. Introduction

The previous chapters have contributed the domain knowledge representation models and the concept navigation models which are constructed consistently in a formal way, using the ontology language, so that they can be integrated and support effectively Web-page recommendations, as experimented in Chapter 7. Therefore, this chapter develops a prototype of the new semantic-enhanced Web-page recommender system (SWRS) utilizing these models to leverage recommendations produced by a community of users to deliver recommendations to an active user. Firstly, the system needs to learn users' Web usage experience which is Web logs of given websites. The Web logs are the records of users' Web browsing behaviours daily, that is Web usage data. By mining Web usage data, useful knowledge can be discovered and represented in the models, i.e. DomainOntoWP or TermNetWP, WPNavNet, and TermNavNet, which can facilitate making Web-page recommendations. Consequently, the system enables users to browse the Web with help from many other users' Web usage experience by recommending them the most interesting and relevant links which might link to the given websites.

This chapter is structured as follows: Section 8.2 firstly introduces an overview of the SWRS. Section 8.3 then proposes the system architecture as well as its sub-systems. Section 8.4 models the static structure of the system. Section 8.5 presents the operation of the system. Section 8.6 illustrates the user interfaces of the system. Finally, Section 8.6 concludes this chapter.

8.2. System Overview

The goal of the proposed SWRS is to target at any websites, and to enable to automate a process from user to user. The process starts from mining Web access sequences requested by users at given websites in a domain of interest for a period of time, to recommending most frequent and relevant pages to users when they browse the Web. To achieve this goal, the developed prototype of the SWRS involves two main tasks: (1) to build the back-end processing components for semantic-enhanced Web-page recommendation, and (2) to develop a front-end component which is a Web browser listing most frequent pages, displaying the Web content, and listing recommended Web-pages.

Moreover, the developed system does not only stop at the one-site level, but can also be applied to more than one websites. The system can admit some Web logs of websites and perform Web-page recommendation for a given current Web access to one certain site. The main point in the recommendation process is to generate a list of Web-pages related to the last visited Web-pages and their domain terms, and these generated pages might belong to relevant websites. The proposed system allows extending the view to other sites rather than one site, which helps to enrich interesting Web-page recommendation results. To obtain the goals, the system employs the proposed models, as presented in the following.

8.2.1. System Statement and Scope

In the developed SWRS, the knowledge is discovered from users' Web usage activities and is processed to be useful for recommendation. Based on the knowledge representation models which have been proposed in the previous chapters, i.e. DomainOntoWP, TermNetWP, TermNavNet, and WPNavNet, the built system consists of five inter-dependent stages as follows.

- *Pre-processing*: Web logs of given websites are collected and cleaned to obtain Web access sequences (WAS) as well as a set of URLs of accessed Web-pages. By crawling the URLs, the titles of Web-pages are fetched.

- *Web usage mining*: the WAS are mined to acquire a complete set of frequent Web access patterns (FWAP), namely Web usage knowledge, using the PLWAP-Mine algorithm (Ezeife & Lu 2005).
- *Domain knowledge construction*: the titles of Web-pages are analysed to extract domain terms in order to build the domain knowledge base of the Web-pages using the DomainOntoWP or TermNetWP models. While DomainOntoWP was constructed semi-automatically for a specific domain, i.e. the MS website, TermNetWP is constructed automatically for any websites.
- *Prediction model*: there are two models built for prediction using ontology language, which are the Web-page navigation model and the domain term navigation model. The Web-page navigation model is proposed in order to automatically generate WPNavNet based on the Web usage knowledge (FWAP). While, the domain term navigation model is proposed in order to automatically generate TermNavNet based on frequent viewed domain term patterns (FVTP) which are generated by integrating FWAP and the domain knowledge, i.e. DomainOntoWP or TermNetWP.
- *Web-page recommendation generation*: there are an option set of making Web-page recommendations by appropriately using the Web-page recommendation strategies with regard to the involved knowledge representation models. For example, given current Web access, the next Web-pages are able to be recommended based on WPNavNet, or a coordination of WPNavNet and DomainOntoWP-based TermNavNet and, or a coordination of WPNavNet and TermNetWP-based TermNavNet.

8.2.2. Major Constraints

It is noted that two major constraints need to be met to be able to obtain recommendation results. Since the system relies on Web usage data which recognizes only visited static Web-pages, Web-pages in the involved websites should be static pages. Moreover, since the domain knowledge of Web-pages is constructed by extracting terms from the title of Web-pages, the content of TITLE tag should be included in the HTML document of each page.

8.3. System Architecture

This section describes the developed system architecture model and five sub-systems.

8.3.1. System Architecture Model

As presented in Section 8.2.1, the system architecture model consists of five main components with respect to the five stages: Pre-processing, Web usage mining, Domain knowledge construction, Prediction model, and Recommendation engine. These components are developed on the back-end side running in background, and can be controlled by an administrator. In addition, to be used by Web users, a Web browser has been developed to display the content of Web-pages and the recommended links on the front-end side. Thus, we have the SWRS architecture, as shown in Figure 8-1.

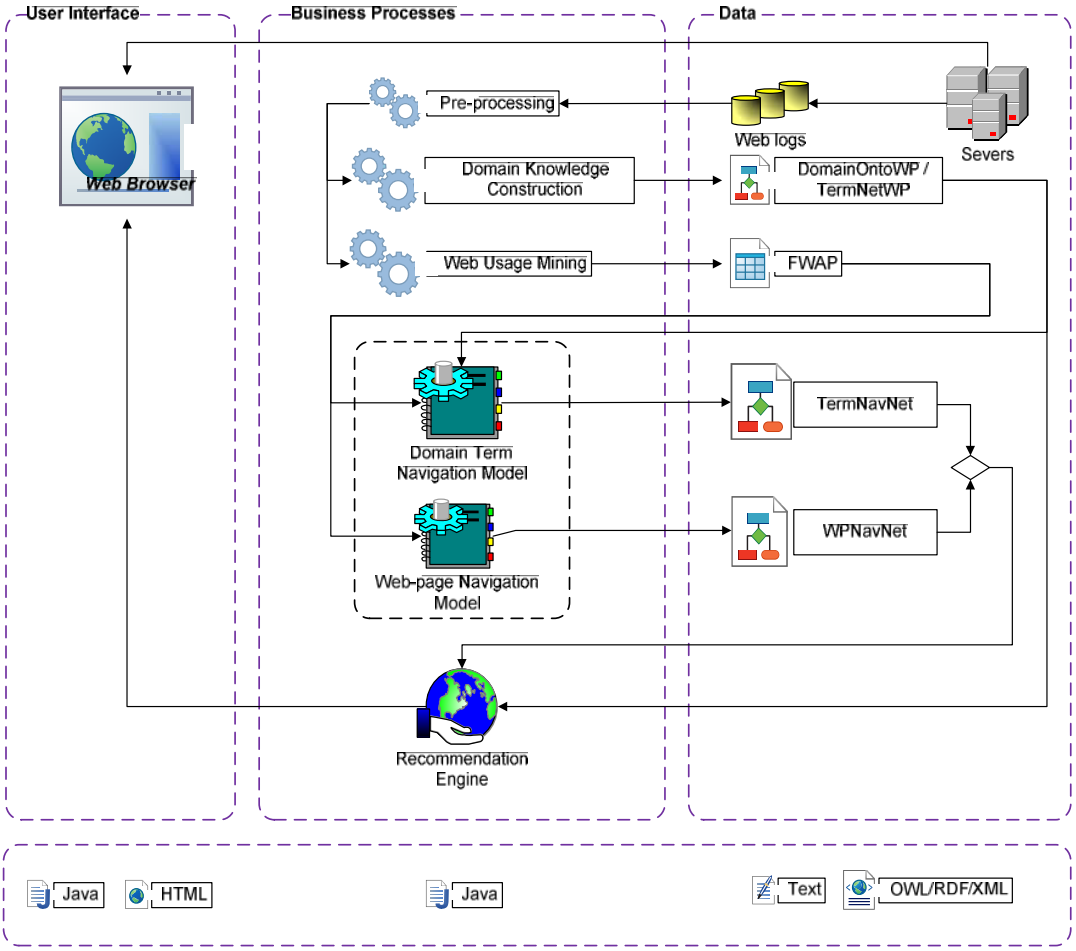


Figure 8-1: Semantic-enhanced Web-page recommender system architecture

It is a three-layer architecture consisting of User interface, Business processes, and Data. Business processes is the main layer embracing the five main components which are Pre-processing, Domain knowledge construction, Prediction model, and Recommendation engine. The Data layer contains source data, i.e. Web logs of servers, processed data, i.e. DomainOntoWP, TermNetWP, FWAP, WPNavNet, and TermNavNet. The User interface layer contains the Web browser. Languages used to develop the system are Java, HTML, OWL/RDF/XML and text. Since Java is a popular programming language useful for algorithm and application implementation thanks to its well-support of various library

package, it is used to implement the control components in the Business processes layer and the User interface layer. The Web log files are basically text files and cleaned to create dataset files in text. FWAP is also presented in text files because it is mined from the Web logs. DomainOntoWP, TermNetWP, WPNavNet, and TermNavNet are presented by OWL/RDF/XML, which is a formal ontology language useful for representing semantic knowledge machine-readable, and facilitating knowledge integration. HTML is used to present Web elements to display Web content on the Web browser.

8.3.2. Sub-System Overview

The components in the system are organized as sub-systems which are developed on the back-end side. Besides, the Web browser on the front-end side is also developed.

1) Back-end

On the back-end side, the five main components of the system are implemented to process data, represent knowledge bases, and control Web-page recommendations.

i. Pre-processing

In the pre-processing sub-system, there are two targets: (1) to acquire datasets of Web access sequences and the URLs of accessed Web-pages; and (2) to retrieve the titles of Web-pages. The input data is Web logs collected from the servers of given websites during a period of time. We use the WebCleaner tool to do the first task. In the second task, Chilkat software package is used to crawl the all URLs and to retrieve the Web-page titles. Table 8-1 describes this sub-system, inputs, outputs, the used sources, and constraints.

Table 8-1: Pre-processing

Description	- Clean Web log data (or Web server access log files) by removing invalid file names and other invalid information - Store data into the data warehouses including tables of users, accessed Web-page paths, Web access sessions, etc. - Create datasets collecting Web access sequences within a period of time - Crawl the accessed Web-page paths to retrieve the Web-page titles
Inputs	Web log files recording users' Web usage sessions of given websites
Outputs	- Tables of users, protocols, paths, and sessions - A dataset of Web access sequences, in which each sequence is a record of Web-pages sequentially visited in a session by a user. - A collection of the URLs and titles of Web-pages
Sources	- Web cleaner: http://sol.cs.uwindsor.ca/~cezeife/webcleaner.tar.gz; Accessed January, 2011 - Web crawler: Chilkat software package (http://www.example-code.com/csharp/spider_simpleCrawler.asp; Accessed March, 2011)
Constraints	- The involved websites are in the same domain of interest.

ii. Web Usage Mining

The target of the Web usage mining sub-system is to obtain FWAP from WAS. The PLWAP-Mine algorithm had been implemented to do this task. A threshold of the minimum support (MinSup) is set to qualify FWAP used in the system. The size of FWAP will be large if the MinSup threshold is small. Therefore, depending on WAS, the MinSup threshold is adjusted to obtain an appropriate quantity of FWAP for knowledge representation in the prediction model stage. Table 8-2 briefly describes the sub-system.

Table 8-2: Web usage mining

Description	Conduct FWAP from WAS with a threshold of minimum support
Inputs	WAS
Outputs	FWAP
Process	PLWAP-Mine

iii. Domain Knowledge Construction

The target of the domain knowledge construction sub-system is to construct the domain knowledge of the accessed Web-pages of the given websites in ontological-style semantic networks. There are two proposed domain knowledge representation models which are the domain ontology model and the semantic network model. The domain ontology model was developed for the MS website, as presented in Chapter 4, and resulted in the domain ontology of Web-pages, namely DomainOntoWP. Since this research only develops the domain ontology of the MS website, this model is applied to a single website, i.e. the MS

website. On the other hand, the semantic network model is more flexible, and can be applied to any websites. With this model, a semantic network of Web-pages, namely TermNetWP, is automatically generated given the Web-page titles retrieved from the involved websites. As a result, we have the domain knowledge base, i.e. TermNetWP, for the all involved websites. Table 8-3 briefly describes the sub-system.

Table 8-3: Domain knowledge construction

Description	Construct the domain knowledge of Web-pages	
Inputs	The collection of the URLs and titles of Web-pages, output from the pre-processing sub-system	
Outputs	DomainOntoWP	TermNetWP
Process	Semi- automatic	Automatic

iv. Prediction Model

The target of the prediction model sub-system is (1) to construct a weighted network of Web-page navigation, namely WPNavNet, and (2) to construct a weighted network of domain term navigation, namely TermNavNet. WPNavNet can be automatically generated given FWAP using the Web-page navigation model, proposed in Chapter 6. TermNavNet can be automatically generated given FWAP and TermNetWP (or DomainOntoWP) using the domain term navigation model, proposed in Chapter 6. The reasoning algorithms which have been proposed for each of the navigation models are also implemented for the preparation of Web-page recommendation in the next stage. Table 8-4 briefly describes the sub-system.

Table 8-4: Prediction model

Description	Construct a weighted network of frequently visited Web-pages	Construct a weighted network of frequently viewed domain terms
Inputs	- FWAP	- FWAP - TermNetWP or DomainOntoWP
Outputs	WPNavNet	TermNavNet
Process	Web-page navigation model	Domain term navigation model

v. Recommendation Engine

The target of the recommendation engine sub-system is to recommend the top-*N* Web-pages which are most popular and relevant to an active user's interest. The recommendation

strategies which have been proposed in Chapter 7 are used in this sub-system. The strategies are classified into three groups, as follows:

(1) Without semantic enhancement: R.PLWAP, R.WP.1st and R.WP.2nd,

(2) With semantic enhancement based on the domain ontology: R.WP.ManTopic.1st.1 and R.WP.ManTopic.2nd.1,

(3) With semantic enhancement based on the semantic network: R.WP.AutoTopic.1st.1, R.WP.AutoTopic.2nd.1, R.WP.AutoTopic.1st.2, R.WP.AutoTopic.2nd.2, R.WP.AutoTopic.1st.3, R.WP.AutoTopic.2nd.3, R.WP.AutoTopic.1st.4, R.WP.AutoTopic.2nd.4, R.WP.AutoTopic.1st.5, R.WP.AutoTopic.2nd.5.

One of the above strategies can be applied to make Web-page recommendations, in which R.WP.AutoTopic.1st.4 and R.WP.AutoTopic.1st.5 are preferred because of their ability to achieve high performance, as observed in the experiments in Chapter 7. Table 8-5 briefly describes inputs, outputs, the strategies used in the Web-page recommendation process, and the used knowledge bases in the sub-system.

Table 8-5: Recommendation engine

Description	Recommend the top-*N* Web-pages		
Inputs	One or two Web-pages last visited by an active user		
Outputs	*N* most interesting and relevant Web-pages		
Process	R.PLWAP, R.WP.1st and R.WP.2nd	R.WP.ManTopic.1st.1 and R.WP.ManTopic.2nd.1	R.WP.AutoTopic.1st.1, R.WP.AutoTopic.2nd.1, R.WP.AutoTopic.1st.2, R.WP.AutoTopic.2nd.2, R.WP.AutoTopic.1st.3, R.WP.AutoTopic.2nd.3, R.WP.AutoTopic.1st.4, R.WP.AutoTopic.2nd.4, R.WP.AutoTopic.1st.5, R.WP.AutoTopic.2nd.5
Knowledge bases	FWAP for PLWAP-Mine		
	WPNavNet for R.WP.1st and R.WP.2nd		
	WPNavNet, DomainOntoWP, and TermNavNet for R.WP.ManTopic.1st.1 and R.WP.ManTopic.2nd.1		
	WPNavNet, TermNetWP, and TermNavNet for R.WP.AutoTopic.1st.1, R.WP.AutoTopic.2nd.1, R.WP.AutoTopic.1st.2, R.WP.AutoTopic.2nd.2, R.WP.AutoTopic.1st.3, R.WP.AutoTopic.2nd.3, R.WP.AutoTopic.1st.4, R.WP.AutoTopic.2nd.4, R.WP.AutoTopic.1st.5, R.WP.AutoTopic.2nd.5		

2) Front-end

On the front-end side, the Web browser is designed to display Web-page recommendation results while a user is accessing the Web-pages of the websites. Table 8-6 briefly describes inputs, outputs, and the process of the Web browser.

Table 8-6: Web browser

Description	List *N* most frequent Web-pages Display the currently visited Web-page Recommend the top-*N* Web-pages based on the last visited pages
Inputs	An accessed Web-page
Outputs	*N* recommended Web-pages
Process	Recommendation engine

8.4. Structural Modelling

This section specifies the static structures of the five sub-systems on the back-end side. For each sub-system, there might be one or more packages. Some main classes for each package will be presented. Especially, this section focuses on the packages in the sub-systems of domain knowledge construction, prediction model, and recommendation engine which are the main contributions of this research.

8.4.1. Pre-processing Sub-System

The input of the pre-processing sub-system is the Web log data. A Web log, or called Web log file or log file, records every request from a user's browser to the Web server. The formats of Web logs are various, depending on the configuration of the Web server. The common log format (CLF or "clog") includes the following seven fields (Markov & Larose 2007b):

- Remote host field,
- Identification field,
- Authuser field,
- Date/time field,
- HTTP request,
- Status code field, and

- Transfer volume field.

Example of a Web log record as follows:

slppp6.intermind.net - - [01/Aug/1995:00:00:10 -0400] "GET /history/skylab/skylab.html HTTP/1.0" 200 1687, where

- Remote host: slppp6.intermind.net,
- Identification: –,
- Authuser: -,
- Date/time: [01/Aug/1995:00:00:10 -0400],
- Request: "GET /history/skylab/skylab.html HTTP/1.0",
- Status code: 200, and
- Transfer volume: 1687.

In order to clean a Web log, the WebCleaner tool, which is considered as the pre-processing package in this sub-system, is used to process the Web log data and to store the processed data into four tables, as described in Figure 8-2 and Table 8-7.

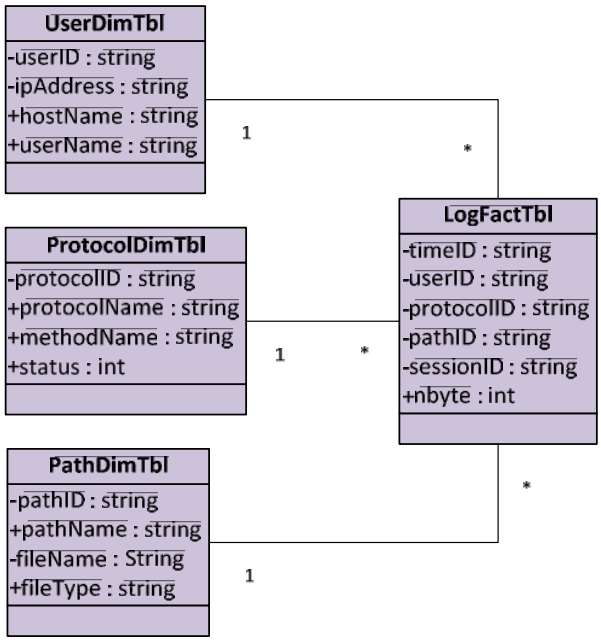


Figure 8-2: ERD storing Web access information of WebCleaner

Table 8-7: Tables in the data warehouse of WebCleaner

Table	Description
ProtocolDimTbl	Web access protocol
PathDimTbl	Information of accessed Web-pages
UserDimTbl	User information
LogFactTbl	Web access events

8.4.2. Web Usage Mining Sub-System

The Web usage mining (WUM) package which is used for the WUM sub-system consists of three classes *PLWAP*, *Node*, and *LinkHeader*, as described in Table 8-8 and Figure 8-3. Class *PLWAP* is used to perform the PLWAP-Mine algorithm (referring to Chapter 3). The main task of PLWAP-Mine algorithm is to construct a PLWAP-tree to generate FWAP from WAS. Class *Node* defines a node, e.g. a page in WAS, in the PLWAP-tree structure. Class *LinkHeader* defines a table linking nodes which have the same event name, e.g. the same page ID, in the PLWAP-tree. Tables 8-9, 8-10, and 8-11 specify the data structures and methods of these three classes.

Table 8-8: The WUM package

Class	Description
PLWAP	Perform the PLWAP-Mine algorithm
Node	Construct a node in the PLWAP-tree
LinkHeader	Construct a table linking nodes with the same event name in the PLWAP-tree

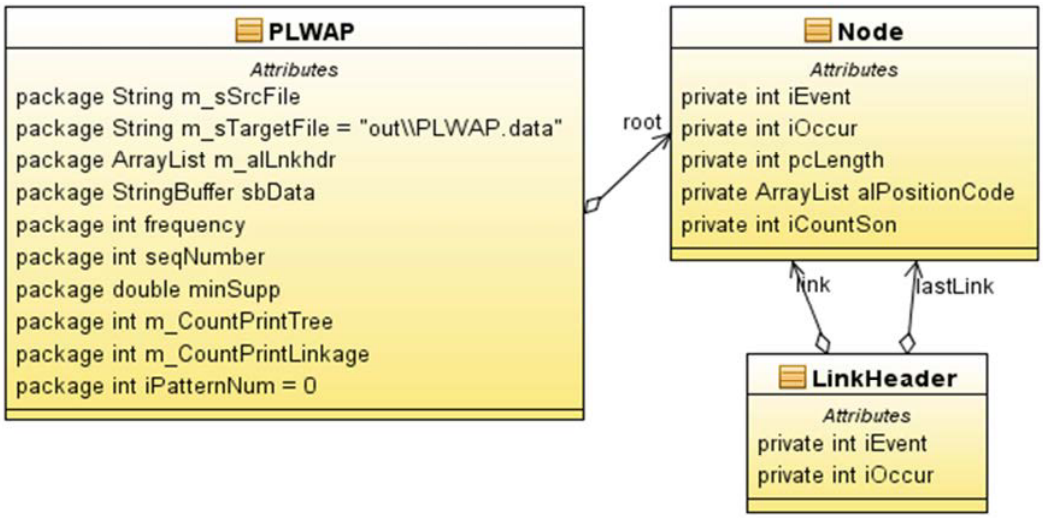


Figure 8-3: The WUM package

Table 8-9: Data structure of class Node in the WUM package

Class Node	Description
- int iEvent	Event name of the node
- int iOccur	Occurrence number of the node
- int pcLength	The length of position code of the node
- ArrayList alPositionCode	The position code is composed of a series of binary digits (32 bits by default). If the length of position code is longer than 32, then we add another 32-bits array.
- int iCountSon	The sum of occurrence number of sons
- Node lSon	The left son node
- Node nextLink	The linkage to a next node having the same event name
- Node parent	The parent node
- Node rSibling	The right sibling node
+ Node(int iEvent_, int iOccur_)	Constructor: constructs a node
+ Node(int iEvent_, int iOccur_, int iCountSon_, int pcLength_, Node parent_, Node lSon_, Node rSibling_, Node nextLink_)	Constructor: constructs a node

Table 8-10: Data structure of class LinkHeader in the WUM package

Class LinkHeader	Description
- int iEvent	Event name of the linkage
- int iOccur	Occurrence number of the event
- Node link	Refers to the first node which occurs in the tree
- Node lastLink	Refers to the last node which occurs in the tree
+ LinkHeader(Node Link_, Node LastLink_, int Event_, int Occur_)	Constructor: constructs a linkage of nodes with the same event name

Table 8-11: Data structure of class PLWAP in the WUM package

Class PLWAP	Description
- String m_sSrcFile	Source data file path, e.g. WAS.data
- String m_sTargetFile = "out\PLWAP.java"	Target file path
- ArrayList m_alLnkhdr	List of *LinkHeaders*
- StringBuffer sbData	String buffer storing the source data
- int frequency	Frequency of a page
- int seqNumber	The number of sequences in WAS
- double minSupp	Minimum support = [0, 1]
- int iPatternNum = 0	The number of discovered FWAP
- Node root	The root node of the PLWAP-tree
+ PLWAP(double minSupport, String SrcFile)	Constructor: performs building a PLWAP-tree given the following parameters - *minSupport*: minimum support - *SrcFile*: source file path
+ StringBuffer RunPLWAP()	Runs the PLWAP-Mine algorithm
+ StringBuffer BuildTree()	Builds the PLWAP-tree Prints the resulting FWAP to the target file
+ MiningProcess(ArrayList rootSet, LinkedList basePattern, int Count)	Finds patterns from the PLWAP-tree - *rootSet*: the set of roots for mining the current tree, every root is a *Node* object. The roots are stored into a list. - *basePattern*: the subsequence which is obtained in the previous loop - *Count*: The sum of occurrence number of suffix tree
+ BuildLinkage(Node start)	Links events with the same label together in the tree.
- ArrayList makeCode(int length, ArrayList pCode, boolean addOne)	Generates a position code for a node - *length*: the length of position code - *pCode*: the position code of its parent or its nearest left sibling - *addOne*: If *addOne* is true, then *pCode* is the position code of its parent, otherwise it is the position code of its nearest left sibling.
- int checkPosition(ArrayList FirstNode, int aLength, ArrayList SecondNode, int dLength)	Checks position between two Nodes - *FirstNode*: the first Node's position code - *aLength*: the length of the first Node 's position code - *SecondNode*: the second Node's position code - *dLength*: the length of the second Node's position code Returns: - 0: if the first Node is the ancestor of the second Node - 1: if the first Node is in the left-tree of the second Node - 2: if the first Node is in the right-tree of the second Node - 3: if the first Node is the descendant of the second Node

8.4.3. Domain Knowledge Construction Sub-System

In this sub-system, there are two packages: (1) Domain Ontology Construction (DOC), and (2) Semantic Network Construction (SNC). The DOC package is developed to support the domain ontology construction of Web-pages, i.e. DomainOntoWP, and to perform the reasoning algorithms on the domain ontology. The SNC package is developed to automatically construct the semantic network of Web-pages, i.e. TermNetWP, and to perform the reasoning algorithms on the semantic network.

(1) ***The DOC package*** consists of three classes *Page*, *OntoPageAdd*, and *Reasoner*, as described in Table 8-12 and Figure 8-4.

Class *Page* describes Web-pages. Since the number of Web-pages at the website is too many to manually map them to respective domain terms in the domain ontology, class *OntoPageAdd* is developed for the domain ontology population with the Web-pages. In this class, method *assignPage_DomainOnto*() performs assigning the Web-pages to respective concept instances if the keywords of Web-page titles respectively match with the terms represented by the instances, according to Algorithm 4-1; methods *checkKeywords*() and *checkSyn*() identify the keywords in a Web-page title and check if they meet the keyword strings in a concept instance, according to the keyword expressions in Table 4-1. Class *Reasoner* performs the reasoning algorithms (Algorithms 4-2 and 4-3), proposed in Chapter 4.

Table 8-12: The DOC package

Class	Description
Page	Describe a Web-page
OntoPageAdd	Perform the domain ontology population, i.e. mapping Web-pages to respective concept instances in the domain ontology automatically
Reasoner	Perform the reasoning algorithms for the domain ontology

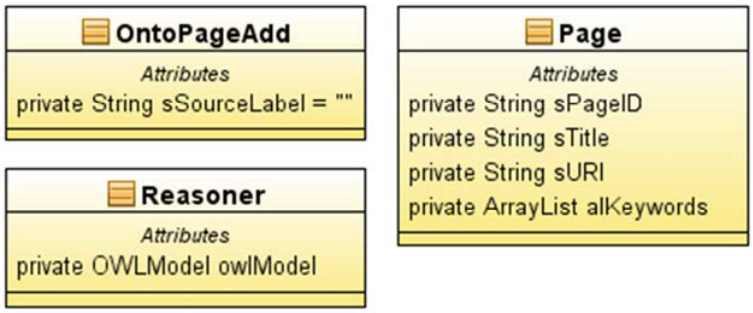


Figure 8-4: The DOC package

Tables 8-13, 8-14, 8-15 present the data structures and methods of classes *Page*, *OntoPageAdd*, and *Reasoner*, respectively.

Table 8-13: Data structure of class Page in the DOC package

Class Page	Description
- String sPageID	The PageID of the page
- String sTitle	The title of the page
- String sURI	The URL of the page
- ArrayList alKeywords	List of keywords in the title
+ Page(String PageID, String Title, String URI)	Constructor: initiates a Web-page object

Table 8-14: Data structure of class OntoPageAdd in the DOC package

Class OntoPageAdd	Description
- String sSourceLabel	Label of source data
+ OntoPageAdd(String ONTOLOGY_URL, String sPath, String sSourceLabel_)	Constructor: populates the domain ontology with Web-pages of the involved websites, given the following parameters: - ONTOLOGY_URL: the path of the domain ontology file - sPath: the path of file containing the collection of the URLs and titles of Web-pages - sSourceLabel: Label of source data
+ Hashtable readTxtFile(String sPath)	Creates Page objects from the text file containing the collection of the URLs and titles of Web-pages sPath: The path of the data file
+ assignPage_DomainOnto(String ONTOLOGY_URL, String sWebPageSrc)	Assigns Pages to respective concept instances in the domain ontology, given the following parameters: - ONTOLOGY_URL: the path of the domain ontology file - sWebPageSrc: the path of file containing the collection of the URLs and titles of Web-pages
+ boolean checkKeywords(String sTitle, Set sKeywords)	Checks if the Title meets the set of keywords, given the following parameters: - sTitle: the title of a Web-page - sKeywords: A set of keywords
+ boolean checkSyn(String sTitle, String sKeyword)	Checks if the title contains a keyword or a synonym of the keyword - sTitle: the title of a Web-page - sKeyword: a keyword

Table 8-15: Data structure of class Reasoner in the DOC package

Class Reasoner	Description
- OWLModel owlModel	The ontology model of DomainOntoWP
+ Reasoner(String sSourceLabel)	Constructor: initiate a reasoner to perform queries over the domain ontology.
+ ArrayList getPages(String sTerm)	Gets a list of PageIDs given a domain term. (referring to Algorithm 4-3 in Chapter 4)
+ ArrayList getTopic(String sPageID)	Gets a set of domain terms (topic) given a Web-page. (referring to Algorithm 4-2 in Chapter 4)
+ getPageTopic(String sPLWAPSrc, String sSrcLabel, String sModeSrcLbl)	Performs generating FVTP from FWAP *sPLWAPSrc*: the file path of FWAP *sSrcLabel*: data source label *sModeSrcLbl*: label assigned to the generated FVTP (Based on Definitions 6.6 and 6.7)

(2) ***The SNC package*** consists of five classes *Instance*, *Page*, *OutLink*, *CollocationMap*, and *Reasoner*, as described in Table 8-16 and Figure 8-5.

As described in the schema of TermNetWP in Chapter 5, classes *Instance*, *WPage*, and *OutLink* represent concepts and relations in TermNetWP in ontology style. In this package, respective Java classes *Instance*, *Page*, and *OutLink* are defined to control operations, such as, populating and reasoning, in TermNetWP. Class *Page* describes a Web-page. Class *Instance* defines a domain term and its properties, such as the number of occurrences of the term, and a list of mapped pages. Class *OutLink* presents an association between two instances with an association weight, i.e. the number of occurrences of the association. In class *OutLink*, method *CheckPhrase*() is used to check if the two associated instances are able to be combined, which is used for the term combination in Algorithm 5-1. Class *CollocationMap* is developed to automatically extract domain terms and generate a TermNetWP from the collection of URLs and titles of Web-pages, according to Algorithm 5-1 and Algorithm 5-2. In class *CollocationMap*, methods *checkForCombination*() and *combineWord2Phrase*() implement the procedures of *check-for-combination* and term combination, respectively, in Algorithm 5-1. Class *Reasoner* performs the reasoning algorithms for TermNetWP, which are presented in Sub-section 5.4.3.

Table 8-16: The SNC package

Class	Description
Page	Describe a Web-page
Instance	Describe a domain term
OutLink	Describe an association between two term instances.
CollocationMap	Construct a TermNetWP
Reasoner	Perform reasoning about domain terms and Web-pages in the TermNetWP

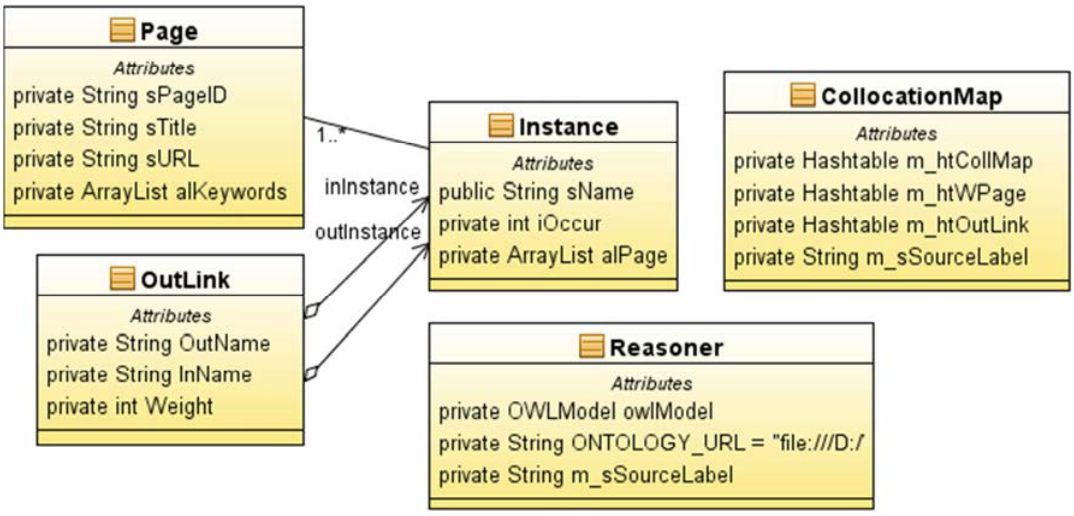


Figure 8-5: The SNC package

Tables 8-17, 8-18, 8-19, 8-20, and 8-21 present the data structures and methods of classes *Page*, *Instance*, *OutLink*, *CollocationMap*, and *Reasoner*, respectively.

Table 8-17: Data structure of class Page in the SNC package

Class Page	Description
- String sPageID	The PageID of the page
- String sTitle	The title of the page
- String sURL	The URL of the page
- ArrayList alKeywords	List of keywords in the title
+ Page(String PageID, String Title, String URL)	Constructor: to initiate a page

Table 8-18: Data structure of class Instance in the SNC package

Class Instance	Description
+ String sName	Instance name, i.e. domain term
- int iOccur	The occurrence number of the instance
- ArrayList alPage	List of pages mapped to the instance
+ Instance(String insName)	Constructor: to initiate an instance named *insName*

Table 8-19: Data structure of class OutLink in the SNC package

Class OutLink	Description
- String InName	The name of the previous term Instance
- String OutName	The name of the next term Instance which occurs after the previous one
- int Weight	Association weight
- Instance inInstance	Refers to the previous term instance
- Instance outInstance	Refers to the next term instance
+ OutLink(Instance ins1, Instance ins2, int iWeight)	Constructor: to construct an *OutLink* associating from *ins1* to *ins2* with the following parameters - *ins1*: inInstance - *ins2*: outInstance - *iWeight*: association weight
+ boolean checkPhrase()	Checks if the two instances of the OutLink are able to be combined

Table 8-20: Data structure of class CollocationMap in the SNC package

Class CollocationMap	Description
- Hashtable m_htCollMap	Table of instances
- Hashtable m_htWPage	Table of Web-pages
- Hashtable m_htOutLink	Table of OutLinks
- String m_sSourceLabel	Label of source data
+ CollocationMap(String sFilePath, String sSourceLabel)	Constructor: to perform the TermNetWP construction given the following parameters - *sFilePath*: the path of file containing the collection of URLs and titles of Web-pages - *sSourceLabel*: Label of source data
+ updateHtOutLink(Instance InInstance, Instance OutInstance)	Updates the information of OutLinks in *m_htOutLink*
+ combineWord2Phrase()	Combines terms which always occur together
+ boolean checkForCombination()	Checks if there are some terms which are able to be combined
+ printOutOnto()	Prints out the *TermNetWP* in the ontology language (OWL)

Table 8-21: Data structure of class Reasoner in the SNC package

Class Reasoner	Description
- OWLModel owlModel	An ontology model of the TermNetWP constructed by CollocationMap
- String ONTOLOGY_URL	The URL of the TermNetWP data source
- String m_sSourceLabel	Source data label
+ Reasoner(String sSourceLabel)	Constructor: initiate a reasoner to perform reasoning about information in the TermNetWP
+ ArrayList getPages(String sDTerm)	Gets a list of PageIDs mapped to a given domain term (*sDTerm*). The resulting pages are sorted in ascending order of the connection weight of a page with the domain term (Referring to Algorithm 5-4 in Chapter 5)
+ int getWeight(ConceptPage.Instance ins, ConceptPage.WPage page)	Gets the connection weight of a page with a term instance *ins*. It is the total of connections from the instance *ins* to the instances which the page mapped to. (Referring to procedure *getWeight* in Algorithm 5-4)
+ ArrayList getPages(ArrayList alDTerms)	Gets pages mapped to a set of domain terms. The resulting pages are sorted in descending order of the number of domain terms which a page is mapped to. (Referring to Algorithm 5-5 in Chapter 5)
+ ArrayList getPages(Hashtable htDTerms)	Gets pages mapped to a set of domain terms with respective domain term prediction probabilities (*htDTerms*). The resulting pages are sorted in descending order of their correlation proportions with the predicted domain term set in *htDTerms*. (Referring to Algorithm 5-6 in Chapter 5)
+ ArrayList getTopic(String sPageID)	Gets a list of domain terms (topic) of a Web-page. The resulting domain terms are sorted in descending order of their occurrence weights. (Referring to Algorithm 5-3 in Chapter 5)
+ Hashtable getDTerms(String sPageID, Double dProb)	Gets a set of domain terms mapped to a Web-page with its prediction probability. (Referring to Algorithm 5-7 in Chapter 5)
+ getPageTopic(String sPLWAPSrc)	Performs generating FVTP from FWAP *sPLWAPSrc*: the file path of FWAP (Based on Definitions 6.6 and 6.7)

8.4.4. Prediction Model Sub-System

In this sub-system, there are two packages: (1) domain term navigation model; and (2) Web-page navigation model. The domain term navigation model package is implemented to represent FVTP, and to result in the semantic Web usage knowledge base, namely TermNavNet. The Web-page navigation model is implemented to represent FWAP, and to result in the transformed Web usage knowledge base in ontology style, namely WPNavNet.

Moreover, the two packages include methods of reasoning about information in the represented Web usage knowledge.

(1) ***The domain term navigation model package*** consists of five classes *Node*, *InLink*, *OutLink*, *NavModel*, and *Reasoner*, as described in Table 8-22 and Figure 8-6.

As described in the schema of domain term navigation model in Chapter 6, classes *cNode* and *cOutLink* represent domain terms and relations, respectively, in TermNavNet in ontology style. In this package, Java classes *Node*, *InLink*, and *OutLink* are defined in order to populate TermNavNet with domain terms and relations. Class *Node* presents *cNode* which describes the state node, i.e. a domain term in TermNavNet. Each *Node,* considered as a current state in TermNavNet, has some *OutLinks* referring to nodes, i.e. next states, and some *InLinks* referring to nodes, i.e. previous states. Class *InLink* describes an association object from a previous state node, e.g. a previously viewed domain term, to the current state node which this object belongs to. Class *OutLink* describes an association object from the current state node, which this object belongs to, to a next state node, e.g. a next viewed domain term. Class *NavModel* implements Algorithm 6-4 to automatically generate a TermNavNet from FVTP based on the schema of domain term navigation model. Class *Reasoner* performs the reasoning algorithms for TermNavNet, i.e. Algorithms 6-5, 6-6, 6-7, and 6-8.

Table 8-22: The domain term navigation model package

Class	Description
Node	Describes a state node in TermNavNet Each *Node*, considered as a current state, has some *OutLinks* referring to nodes, i.e. next states, and some *InLinks* referring to nodes, i.e. previous states.
InLink	Describes an association from a previous state node, e.g. a previously viewed domain term, to a state current node. It refers to the previous state node of the current state node which it belongs to.
OutLink	Describes an association from a current state node to a next state node, e.g. a next viewed domain term, with a transition probability. It refers to the next state node of the current state node which it belongs to.
NavModel	Constructs TermNavNet using Algorithm 6-4
Reasoner	Perform reasoning about the predicted next domain terms in TermNavNet

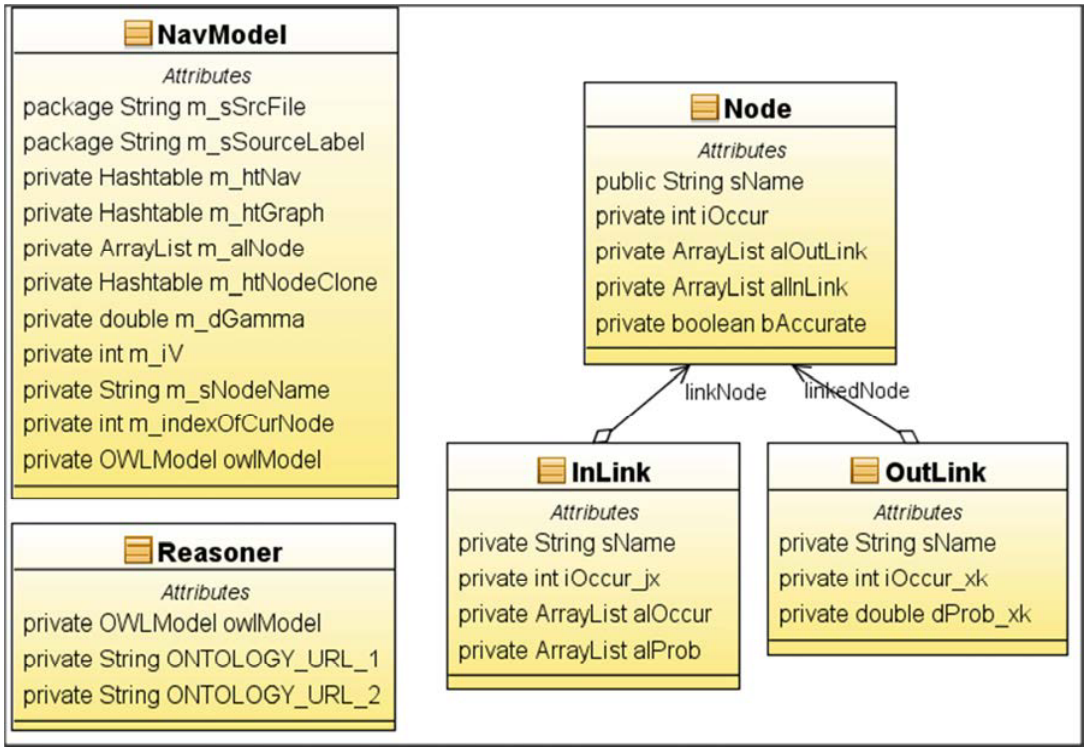


Figure 8-6: The domain term navigation model package

Tables 8-23, 8-24, 8-25, 8-26, and 8-27 specify the data structures and methods of the five classes in the domain term navigation model package. This package is implemented according to the method of building the dynamic clustering-based Markov model in (Borges & Levene 2004). In order to obtain second-order prediction accuracy in the model, *Node* objects might be cloned if they are not accurate, that is, if the difference between their first and second-order probabilities is greater than an accuracy threshold, e.g., $\gamma = 0$. Table 8-23 presents class *Node* including essential methods of cloning a state node in TermNavNet. Method *Run_KMean*() runs the K-mean algorithm to build the clones of the node. Methods *Check_Accurate*() and *test_Accurate*() are used to check the accuracy of the state node. *InLink* objects of an inaccurate state node are clustered with respect to state nodes cloned from the inaccurate state node. Methods *CloningState*() and *Clone*() perform cloning the state node.

Table 8-23: Data structure of class Node in the domain term navigation model package

Class Node	Description
- String sName	Node name
- int iOccur	Occurrence count
- ArrayList alOutLink	List of *OutLink* objects
- ArrayList alInLink	List of *InLink* objects
- boolean bAccurate	A flag: to check if the state node is accurate or not
+ Node(String sName_, int iOccur_)	Constructor: to construct a state node
+ Graph.Node Clone()	Clones the *Node* object
+ boolean Check_Accurate(int iV, double dGamma)	Checks the accuracy of the state node (Referring to the definition of state accuracy in (Borges & Levene 2004))
+ ArrayList Run_KMean(int iV, double dGamma)	Runs the K-Mean algorithm to result in the clones of the node (Referring to the clustering-based cloning method in (Borges & Levene 2004))
- boolean test_Accurate(ArrayList alSubIn_link, int iV, double dGamma)	Checks if the state node is accurate based on a cluster of *inLinks* (Referring to the definition of state accuracy in (Borges & Levene 2004))
+ ArrayList CloningState(ArrayList alIn_link)	Clones the inaccurate state nodes given the clusters of *inLinks* (referring the definition of state cloning in (Borges & Levene 2004))

Table 8-24: Data structure of class InLink in the domain term navigation model package

Class InLink	Description
- String sName	*InLink* name being the name of a previous state node
- Node linkNode	Refers to the previous state node
- int iOccur_jx	The occurrence count means the number of times this previous state node occurs before the current state node which this *InLink* object belongs to
- ArrayList alOccur	The list of occurrence counts for computing second-order prediction probabilities at the current state node which this *InLink* object belongs to
- ArrayList alProb	The occurrence probabilities for computing second-order prediction probabilities at the current state node which this *InLink* object belongs to
+ InLink(String sName_, Node node, int iOccur_jx_)	Constructor: to construct an *InLink* object
+ InLink Clone()	Clones the *InLink* object

Table 8-25: Data structure of class OutLink in the domain term navigation model package

Class OutLink	Description
- String sName	*OutLink* name being the name of a next state node
- Node linkedNode	Refers to the next state node
- int iOccur_xk	The occurrence count means the number of times this next state node occurs after the current state node which this *OutLink* object belongs to
- double dProb_xk	The occurrence probability
+ OutLink(String sName_, Node linkedNode_, int iOccur_xk_)	Constructor: to construct an *OutLink* object
+ OutLink Clone()	Clones the *OutLink* object

Table 8-26: Data structure of class NavModel in the domain term navigation model package

Class NavModel	Description
- String m_sSrcFile	Source file path, e.g. "./in/PLWAP.data"
- String m_sSourceLabel	Source data label
- Hashtable m_htNav	Table of state nodes
- Hashtable m_htGraph	Table of traversed node names
- ArrayList m_alNode	List of nodes, used for sorting
- Hashtable m_htNodeClone	Table of cloned nodes
- double m_dGamma	An accuracy threshold parameter (γ)
- int m_iV	A parameter (V) stating the number of times a state has to be visited for the estimation of its probabilities is considered reliable. E.g, $m\_iV = 5$
- OWLModel owlModel	Ontology model of TermNavNet
+ NavModel(double dGamma, int iV, String SrcFile, String sSourceLabel)	Constructor: to initiate a navigation network with parameters Gamma (γ) and V Given a source file (*SrcFile*) (Referring to the method of building the dynamic clustering-based Markov model in (Borges & Levene 2004))
+ createNavModel(boolean bPrintout1stModel)	Creates a first-order navigation network
- update_Prob_xk()	Updates second-order probabilities of the model
+ createMixNavModel()	Creates a higher-order navigation network
- ArrayList sortNodes()	Sorts the state nodes in order by occurrences
- compute_2nd_prob()	Computes second-order probabilities
+ printOutOnto()	To print out the navigation network in the ontology language (OWL)

Table 8-27: Data structure of class Reasoner in the domain term navigation model package

Class Reasoner	Description
- OWLModel owlModel	Ontology model of TermNavNet
- String ONTOLOGY_URL_1	The URL of ontology data source 1 for the first-order navigation network
- String ONTOLOGY_URL_2	The URL of ontology data source 2 for the second-order navigation network
+ Reasoner(String sSourceLabel, boolean b1stModelmode)	Constructor: to initiate a reasoner to perform queries over the TermNavNet. If *b1stModelmode* is true, then the ontology data source 1 is used, else the ontology data source 2 is used
+ ArrayList reasoning(String sPreDTerm, String sCurDTerm)	Infers next domain terms for a given previous domain term (*sPreDTerm*) and a given current domain term (*sCurDTerm*) Return domain terms in order by prediction probability (Referring to Algorithm 6-5)
+ ArrayList reasoning(String sCurDTerm)	Infers next domain terms for a given current domain term (*sCurDTerm*) Return domain terms in order by prediction probability (Referring to Algorithm 6-6)
+ Hashtable reasoningTopProb(String sPreDTerm, String sCurDTerm)	Infers next domain terms for a given previous domain term (*sPreDTerm*) and a given current domain term (*sCurDTerm*) Return domain terms along with the corresponding prediction probabilities (Referring to Algorithm 6-7)
+ Hashtable reasoningTopProb(String sCurDTerm)	Infers next domain terms for a given current domain term (*sCurDTerm*) Return domain terms along with the corresponding prediction probabilities (Referring to Algorithm 6-8)

(2) ***The Web-page navigation model package*** is similar to the domain term navigation model package and consists of five classes *Node*, *InLink*, *OutLink*, *NavModel*, and *Reasoner*, as described in Table 8-28. However, instead of representing domain terms in the domain term navigation model, this package is used to represent Web-pages in the Web-page navigation model.

As described in the schema of Web-page navigation model in Chapter 6, classes *cNode* and *cOutLink* represent Web-pages and relations in WPNavNet in ontology style. In this package, Java classes *Node*, *InLink*, and *OutLink* are defined in order to populate WPNavNet with pages and relations. Class *Node* presents *cNode* which describes the state node, i.e. a page in WPNavNet. Each *Node,* considered as a current state in WPNavNet, has some *OutLinks* referring to nodes, i.e. next states, and some *InLinks* referring to nodes, i.e.

previous states. Class *InLink* describes an association object from a previous state node, e.g. a previously visited page, to the state current node which this object belongs to. Class *OutLink* describes an association object from the current state node, which this object belongs to, to a next state node, e.g. a next visited page. Class *NavModel* implements Algorithm 6-1 to automatically generate a WPNavNet from FWAP based on the schema of Web-page navigation model. Class *Reasoner* performs the reasoning algorithms for WPNavNet, i.e. Algorithms 6-2, and 6-3.

Table 8-28: The Web-page navigation model package

Class	Description
Node	Describes a state node in WPNavNet Each *Node*, considered as a current state, has some OutLinks referring to nodes, i.e. next states, and some InLinks referring to nodes, i.e. previous states.
InLink	Describes an association from a previous state node, e.g. a previously visited page, to a state current node. It refers to the previous state node of the current state node which it belongs to.
OutLink	Describes an association from a current state node to a next state node, e.g. a next visited page, with a transition probability. It refers to the next state node of the current state node which it belongs to.
NavModel	Constructs WPNavNet using Algorithm 6-1
Reasoner	Perform reasoning out the predicted next pages in WPNavNet

8.4.5. Recommendation Engine Sub-System

The Recommendation Engine (RE) package which is developed for this sub-system consists of one main class *RecommendationStrategies*. Class *RecommendationStrategies* performs recommendation as well as the evaluation of Web-page recommendations based on the different strategies which have been presented in Chapter 7.

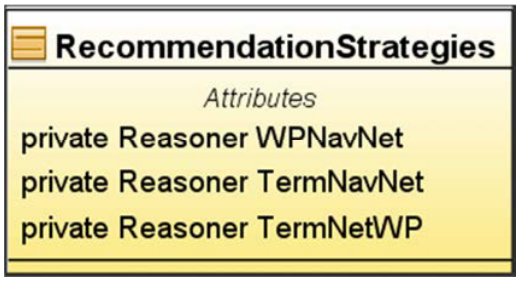


Figure 8-7: The RE package

For instance, Figure 8-7 and Table 8-29 specify the data structure and methods of class *RecommendationStrategies* in case using WPNavNet, TermNetWP and TermNavNet for Web-page recommendation. Method *ConceptualWebRecommendation*() recommends the top-*N* next Web-pages given a previous page and a current page using one of the proposed Web-page recommendation strategies. The *maxRecLength* parameter value is the recommendation length, i.e., *N*. Method *WR_Evaluation*() implements Algorithm 7-8 to evaluate the used strategy.

Table 8-29: The RE package

Class RecommendationStrategies	Description
- Graph.Reasoner WPNavNet	The reasoner for the used WPNavNet
- Graph.Reasoner TermNavNet	The reasoner for the used TermNavNet
- CollocationMap.Reasoner TermNetWP	The reasoner for the used TermNetWP
+ RecommendationStrategies(String sSourceLabel, String sWASSrc, String sPLWAPSrc, String sSourceModelLabel, String sSourceWPModelLabel, boolean b1stModelMode)	Constructor: to initiate reasoner objects for recommendation If *b1stModelMode* is true, then the recommendation is based on the first-order navigation network, else based on the second-order navigation network
+ List ConceptualWebRecommendation(String sPrePageID, String sPageID, int mode, int maxRecLength)	Recommend next pages given - *sPrePageID*: the PageID of a previous page - *sPageID*: the PageID of a current page - *mode*: recommendation mode, e.g. if mode =1: if *b1stModelMode* is true then uses R.WP.AutoTopic.1st.1, else R.WP.AutoTopic.2nd.1 =2: if *b1stModelMode* is true then uses R.WP.AutoTopic.1st.2, else R.WP.AutoTopic.2nd.2 … (referring to the Web-page recommendation algorithms in Chapter 7) - *maxRecLength*: the recommendation length
+ WR_Evaluation(String WASFile, String sPLWAPSrc, String sSourceLabel, int mode, int maxRecLength)	Evaluates Web-page recommendation given testing data (*WASFile*) *sPLWAPSrc*: the path of a file containing FWAP

In similar ways, we can make Web-page recommendations based on DomainOntoWP or WPNavNet by changing the reasoner objects appropriately.

8.5. Operation

This section describes the operation of the five sub-systems on the back-end side in the proposed recommender system by an experimental example. The tasks of knowledge discovery and representation are carried out automatically through the system. The process starts from collecting users' Web usage activities at given websites, which are logged into the Web servers on a daily basis, namely Web log files. This Web usage data is pre-processed to obtain useful information for mining and representing knowledge for Web-page recommendation. The knowledge bases which are built by using the automatic approaches to the knowledge representation models, i.e., TermNetWP and TermNavNet, are taken into account in this section.

The following clarifies how data is processed at the five stages with respect to the five sub-systems by giving sample input and output data. Website *handbook.uts.edu.au* is used to demonstrate sample data.

8.5.1. Pre-processing

In the pre-processing stage, the WebCleaner tool is firstly used to filter Web log data. Figure 8-8 describes a few records in the Web log file of website *handbook.uts.edu.au*.

```
212.184.196.214 - - [19/Apr/2011:16:03:29 +1000] "GET /bus/index.html HTTP/1.0" 200 12737
203.39.178.154 - - [19/Apr/2011:16:03:30 +1000] "GET /course_areas.html HTTP/1.1" 200 12271
138.25.225.162 - - [19/Apr/2011:16:03:30 +1000] "GET /courses/c04245.html HTTP/1.1" 200 26938
110.33.97.31 - - [19/Apr/2011:16:03:30 +1000] "GET /css/override.css HTTP/1.1" 200 10415
212.184.196.214 - - [19/Apr/2011:16:03:34 +1000] "GET /bus/lists/alpha.html HTTP/1.0" 200 62922
110.33.97.31 - - [19/Apr/2011:16:03:36 +1000] "GET /images/css/body_bg2.png HTTP/1.1" 200 18682
110.33.97.31 - - [19/Apr/2011:16:03:36 +1000] "GET /images/css/topbg.gif HTTP/1.1" 200 769
110.33.97.31 - - [19/Apr/2011:16:03:36 +1000] "GET /images/css/handbook-banner-new.jpg HTTP/1.1" 200 37193
110.33.97.31 - - [19/Apr/2011:16:03:36 +1000] "GET /images/css/triangle-contact.png HTTP/1.1" 200 1008
124.188.226.68 - - [19/Apr/2011:16:03:38 +1000] "GET /subjects/91400.html HTTP/1.1" 200 14561
212.184.196.214 - - [19/Apr/2011:16:03:40 +1000] "GET /subjects/25923.html HTTP/1.0" 200 14321
```

Figure 8-8: Sample Web log of website *handbook.uts.edu.au*

The Web log data is cleaned by the WebCleaner tool and stored into the four tables *UserDimTbl*, *ProtocolDimTbl*, *PathDimTbl*, and *LogFactTbl*. For example, Tables 8-30, 8-31, 8-32, and 8-33 show the sample output data in the four tables, as follows.

Table 8-30: Sample data in Table *UserDimTbl*

UserDimTbl			
userID	**ipAddress**	**hostName**	**userName**
1	212.184.196.214	-	-
2	203.39.178.154	-	-
3	138.25.225.162	-	-
4	59.167.240.32	-	-

Table 8-31: Sample data in Table *ProtocolDimTbl*

ProtocolDimTbl			
protocolID	**protocolName**	**methodName**	**status**
1	HTTP/1.0	GET	200
2	HTTP/1.1	GET	200
3	HTTP/1.1	GET	404
4	HTTP/1.0	GET	404

Table 8-32: Sample data in Table *PathDimTbl*

PathDimTbl			
pathID	**Pathname**	**fileName**	**fileType**
1	/bus	index	html
2	/	course_areas	html
3	/courses	c04245	html
4	bus/lists	alpha	html
5	/courses	c07119	html

Table 8-33: Sample data in Table *LogFactTbl*

LogFactTbl					
timeID	**userID**	**protocolID**	**pathID**	**sessionID**	**nbyte**
19/Apr/2011:16:03:29/+1000	1	1	1	1	12737
19/Apr/2011:16:03:30/+1000	2	2	2	2	12271
19/Apr/2011:16:03:30/+1000	3	2	3	3	26938
19/Apr/2011:16:03:34/+1000	1	1	4	1	62922
19/Apr/2011:16:03:46/+1000	4	2	5	4	17992
19/Apr/2011:16:03:49/+1000	5	2	6	5	6826
19/Apr/2011:16:03:49/+1000	6	2	7	6	12738
19/Apr/2011:16:03:52/+1000	7	2	8	7	13819
19/Apr/2011:16:03:57/+1000	5	2	9	5	6426
19/Apr/2011:16:03:34/+1000	1	1	4	1	62922

From the four tables, a dataset can be created for a period of time, e.g. 27-28/Apr/2011, and saved as a *.suv* file. Figure 8-9 shows the sample dataset which is created from the output tables by using WebCleaner.

userID	sessionID	Web-page visited
6	20896	117
6	21758	7
8	21163	7 116 687 42
8	21223	1 163 127 130 131 196 199 375
8	21391	7 116 81
8	21512	1
8	21618	7 116 554
9	20941	483 17 483 17 17 483 2 277
9	21544	450
…		

Figure 8-9: Sample dataset obtained from the Web log of website *handbook.uts.edu.au*

The sequences of visited Web-pages in the dataset are WAS, in which each figure refers to a PageID. In addition, the WebCrawler software package is used to crawl the paths of the accessed Web-pages based on Table *PathDimTbl*, which records in turn Web-pages accessed by users in the period of time. In the context of this example, the system retrieves only the Web-page titles which contain the domain terms of the first 700 accessed Web-pages. Table 8-34 describes a sample collection of the titles and URLs of accessed Web-pages.

Table 8-34: A sample collection of the titles and URLs of accessed Web-pages

PageID	Title	URL
1	UTS: Business - UTS Handbook	http://www.handbook.uts.edu.au/bus/index.html
2	UTS: Course areas - UTS Handbook	http://www.handbook.uts.edu.au//course_areas.html
3	UTS: C04245v1 Master of Arts in Teaching English to Speakers of Other Languages - Education, UTS Handbook	http://www.handbook.uts.edu.au/courses/c04245.html
4	UTS: Alphabetical list of subjects - Business, UTS Handbook	http://www.handbook.uts.edu.au/bus/lists/alpha.html
5	UTS: C07119v1 Graduate Diploma in Design - Design, Architecture and Building, UTS Handbook	http://www.handbook.uts.edu.au/courses/c07119.html
6	UTS: Handbook 2008 - Index	http://www.handbook.uts.edu.au/2008/order_form.html
7	UTS: Law - UTS Handbook	http://www.handbook.uts.edu.au/law/index.html
8	UTS: General information - UTS Handbook	http://www.handbook.uts.edu.au/general/index.html
9	UTS: Handbook 2008 - Business	http://www.handbook.uts.edu.au/2008/bus/index.html
10	UTS: C04150v4 Master of Industrial Property - Law, UTS Handbook	http://www.handbook.uts.edu.au/courses/c04150.html

In case of multiple sites, each Web log file of a website is pre-processed. The PageIDs of Web-pages of the all websites are renamed and united to form a collection of the titles and URLs as Table 8-34. Afterwards, datasets which are created from the Web log of each website are combined into a set of WAS for the next processes.

8.5.2. Web Usage Mining

As mentioned in the WUM sub-system, in the WUM stage, the PLWAP-Mine algorithm is used to discover FWAP from WAS. A minimum support is applied to qualify the discovered FWAP. Table 8-35 demonstrates a sample data of WAS including three columns: Order number (#), Number of pages (No.P), and Sequence of visited pages (Seq.P); and a sample FWAP with MinSup = 0.6%.

Table 8-35: A sample FWAP discovered from a sample WAS with MinSup = 0.6 %

WAS						FWAP (MinSup = 0.6%)			
#	No.P	Seq.P				15			
11	1	117				15	15		
12	1	7				15	15	15	
13	1	7				15	15	127	
14	4	7	116	687	42	15	15	127	127
15	8	1	163	127	130	15	24		
	131	196	199	375		15	27		
16	3	7	116	81		15	45		
17	1	1				15	57		
18	3	7	116	554		15	57	127	
19	8	483	17	483	17	15	73		
	17	483	2	277		15	74		
20	1	450				…			
…									

8.5.3. Domain Knowledge Construction

In this stage, the model of automatic semantic network construction is involved. A semantic network of Web-pages is populated by using the TermNetWP model. The schema of the TermNetWP model is documented in OWL, and is then fed into Protégé to generate the Java schema of TermNetWP. Based on this Java schema, an OWL document of TermNetWP can be automatically generated given domain terms extracted from the titles of

Web-pages. Figure 8-10 depicts a sample TermNetWP obtained from the 700 pages in website *handbook.uts.edu.au*, in the Protégé interface. There are 775 term instances and 1934 association relations (*OutLink*) between the term instances. The *iOccur*, *fromOutLink*, *hasOutLink* and *hasWPage* properties of class *Instance* have been explained in the schema of TermNetWP in Chapter 5. Each *OutLink* instance is presented by a combination of two terms in the order they appear in titles. For instance, term "Business" occurs 118 times, is followed by terms, such as "Engineering", "Finance", and "Honours", follows terms, such as "CBK90100", "CBK90150", and "Diploma", and has Web-pages, such as *utshb_1*, *utshb_103*, and *utshb_105*.

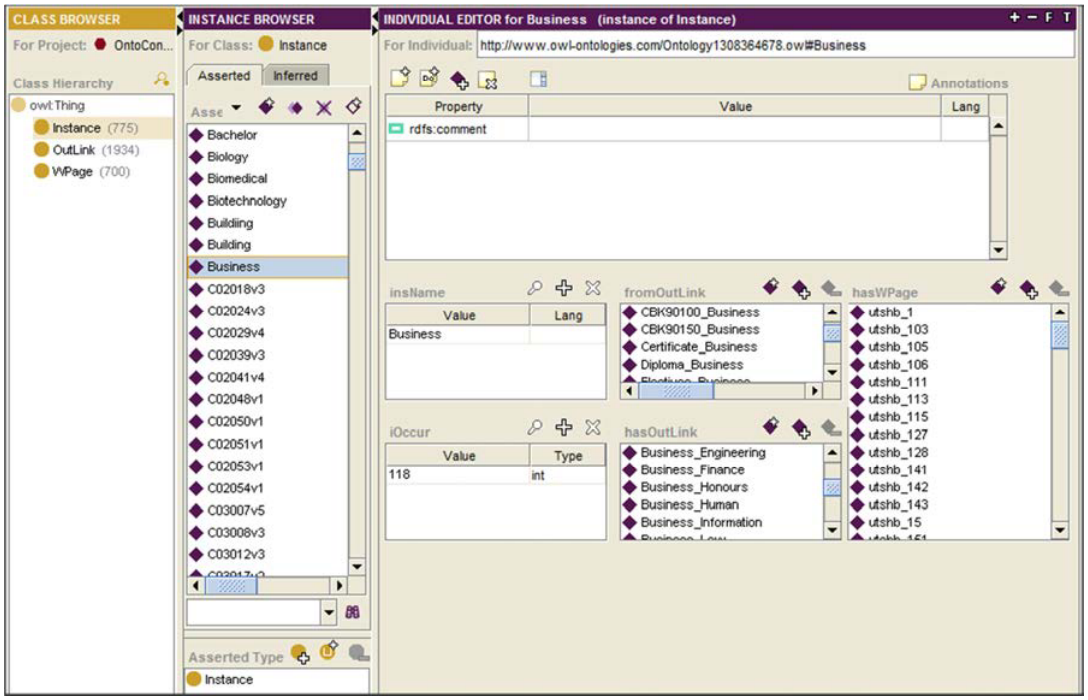


Figure 8-10: Sample TermNetWP in the Protégé interface

8.5.4. Prediction Model

In the prediction model stage, the domain term navigation model is involved. It is used to automatically generate a TermNavNet given the FWAP. The schema of domain term navigation model is documented in OWL, and is then fed into Protégé to generate the Java

schema of domain term navigation model. Based on this Java schema, an OWL document of TermNavNet can be populated. Figure 8-11 depicts a sample 1st-order TermNavNet in the Protégé interface. In class *cNode*, the *sCInLink* property refers to the names of previously viewed domain terms. The *hasCOutLink* property refers to c*OutLink* instances whose names present transitions from a domain term to another one. The *LinkedcNode* property is not used in this case. The 1st-order TermNavNet is obtained from FVTP which is a result of integrating of FWAP with TermNetWP. In this sample, there are 245 frequently viewed domain terms and 3087 transitions (c*OutLink*) from domain term to domain term. For instance, domain term "Business" is frequently viewed after domain terms, such as "Media", "Studies", and "Nursing", and before domain terms, such as "Accounting", "Administration", and "Architecture".

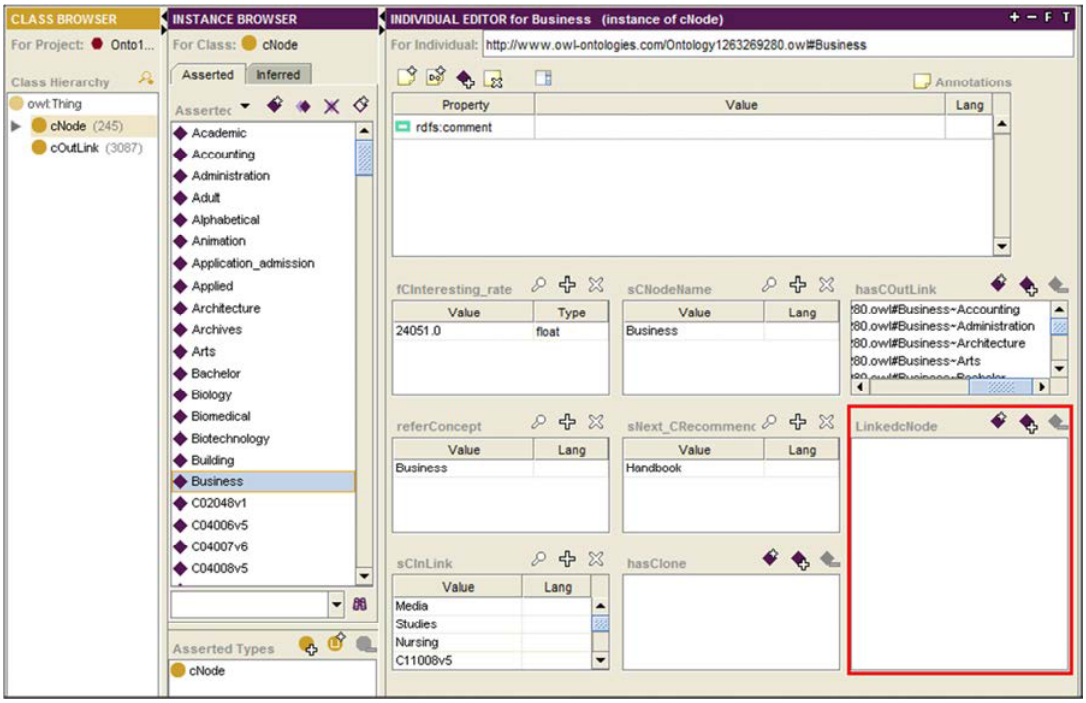


Figure 8-11: Sample TermNavNet in the Protégé interface

8.5.5. Web-page Recommendation

The Web-page recommendation is performed based on the knowledge bases represented in the previous stages. In this sub-section, TermNetWP and TermNavNet are used to make Web-page recommendations. When a user visits a Web-page, a Web-page request is sent to the system to identify the currently visited page. The last two visited pages are taken into account to recommend Web-pages which are most frequent and relevant.

Figures 8-12 and 8-13 describe respectively the two examples of recommending the top-10 Web-pages. In the first example, it is assumed that page *utshb_719* titled "UTS: SMJ02036 Business Information Systems - UTS Handbook" is visited first, and the system recommends ten Web-pages, such as, *utshb_351*, *utshb_127*, and *utshb_468*, about courses in the Bachelor of Business; *utshb_579*, about a course in the Bachelor of Information Technology (IT); *utshb_383* and *utshb_61*, about the Business and IT handbook, respectively; *utshb_88* and *utshb_163*, about postgraduate and undergraduate courses in Business; and *utshb_426* and *utshb_542*, about postgraduate and undergraduate courses in IT. Generally, all recommended pages are about courses and information in Business and IT because the current page is a sub-major of information systems for business.

```
Previous page: Start [Start]
Current page: utshb_719 [SMJ02036, Information, Systems, UTS, Business, Handbook]
Recommended pages:
Page: utshb_351 [C10027v4, UTS, Bachelor, Business, Handbook]
Page: utshb_127 [C10026v4, UTS, Bachelor, Business, Handbook]
Page: utshb_468 [C10226v2, UTS, Bachelor, Business, Handbook]
Page: utshb_579 [UTS, Information, Technology, C10143v5, Bachelor, Handbook]
Page: utshb_383 [UTS, Business, Handbook]
Page: utshb_88 [Postgraduate, courses, UTS, Business, Handbook]
Page: utshb_61 [UTS, Information, Technology, Handbook]
Page: utshb_426 [Information, Postgraduate, UTS, Technology, courses, Handbook]
Page: utshb_542 [Information, Undergraduate, Technology, courses, UTS, Handbook]
Page: utshb_163 [Undergraduate, UTS, courses, Business, Handbook]
```

Figure 8-12: Web-page recommendation results of page utshb_719

In the second example, if page *utshb_127* titled "UTS: C10026v4 Bachelor of Business - Business, UTS Handbook" is visited after *utshb_719*, the system will recommend ten Web-pages, such as, *utshb_127*, *utshb_468*, *utshb_351*, and *utshb_159*, about courses in

Business; *utshb_163*, *utshb_383*, and *utshb_88*, about the common information of courses in Business; *utshb_306* about the sub-major of operations theory and management in IT; *utshb_258* about courses in Engineering; and *utshb_78* about courses in Communication. As a whole, more pages of Business are recommended, and all of the recommended pages are relevant to the visited pages.

Previous page: utshb_719 [SMJ02036, Information, Systems, UTS, Business, Handbook]
Current page: utshb_127 [C10026v4, UTS, Bachelor, Business, Handbook]
Recommended pages:
Page: utshb_127 [C10026v4, UTS, Bachelor, Business, Handbook]
Page: utshb_468 [C10226v2, UTS, Bachelor, Business, Handbook]
Page: utshb_351 [C10027v4, UTS, Bachelor, Business, Handbook]
Page: utshb_163 [Undergraduate, UTS, courses, Business, Handbook]
Page: utshb_159 [C10026v3, 2010, Bachelor, Business, UTS, Handbook]
Page: utshb_383 [UTS, Business, Handbook]
Page: utshb_306 [SMJ02056, Operations, Theory, Management, UTS, Handbook]
Page: utshb_258 [Undergraduate, UTS, courses, Engineering, Handbook]
Page: utshb_88 [Postgraduate, courses, UTS, Business, Handbook]
Page: utshb_78 [Undergraduate, courses, UTS, Handbook, Communication]

Figure 8-13: Web-page recommendation results of page utshb_719 and utshb_127

In sum, there may be not many recommended pages which are specifically related to the visited pages because this example involves the first accessed 700 Web-pages in the period of time, which do not cover all Web-pages of course descriptions. In practice, it is necessary to filter accessed Web-pages to obtain more specific Web-pages of interest in order to improve recommendation results.

8.6. Interface Description

This section describes the user interfaces of the developed recommender system, in which the sub-system interfaces of Pre-processing, Web usage mining, Domain knowledge construction using the TermNetWP model, named Semantic network construction, Prediction model using the domain term navigation model, named Conceptual prediction model, Recommendation engine, and Web browser are shown. Figure 8-14 depicts the main frame of the system including the five functions with respect to the five stages mentioned earlier.

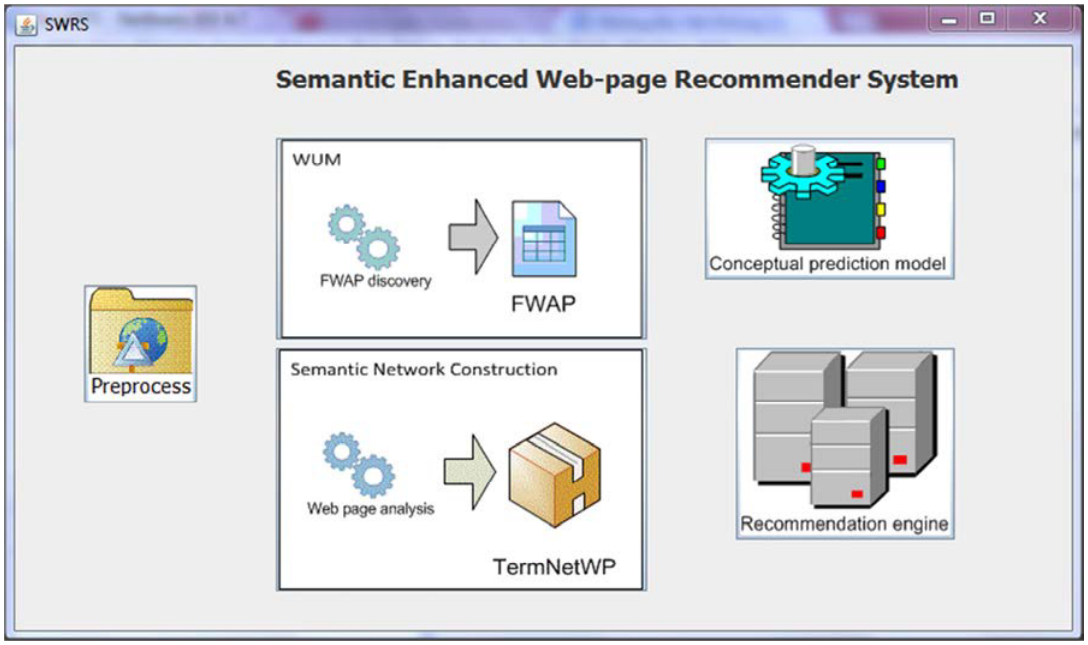


Figure 8-14: Main frame of the semantic-enhanced Web-page recommender system

The following presents the interfaces of the five functions on the back-end side and the Web browser on the front-end side.

8.6.1. Back-end

The function interfaces on the back-end side are presented, as follows.

(1) Pre-processing

In the pre-processing interface, there are three functions: (1) cleaning a Web log using WebCleaner to create datasets, (2) refining the dataset to obtain WAS, (3) crawling all the Web-page paths to retrieve titles using WebCrawler, as illustrated in Figure 8-15.

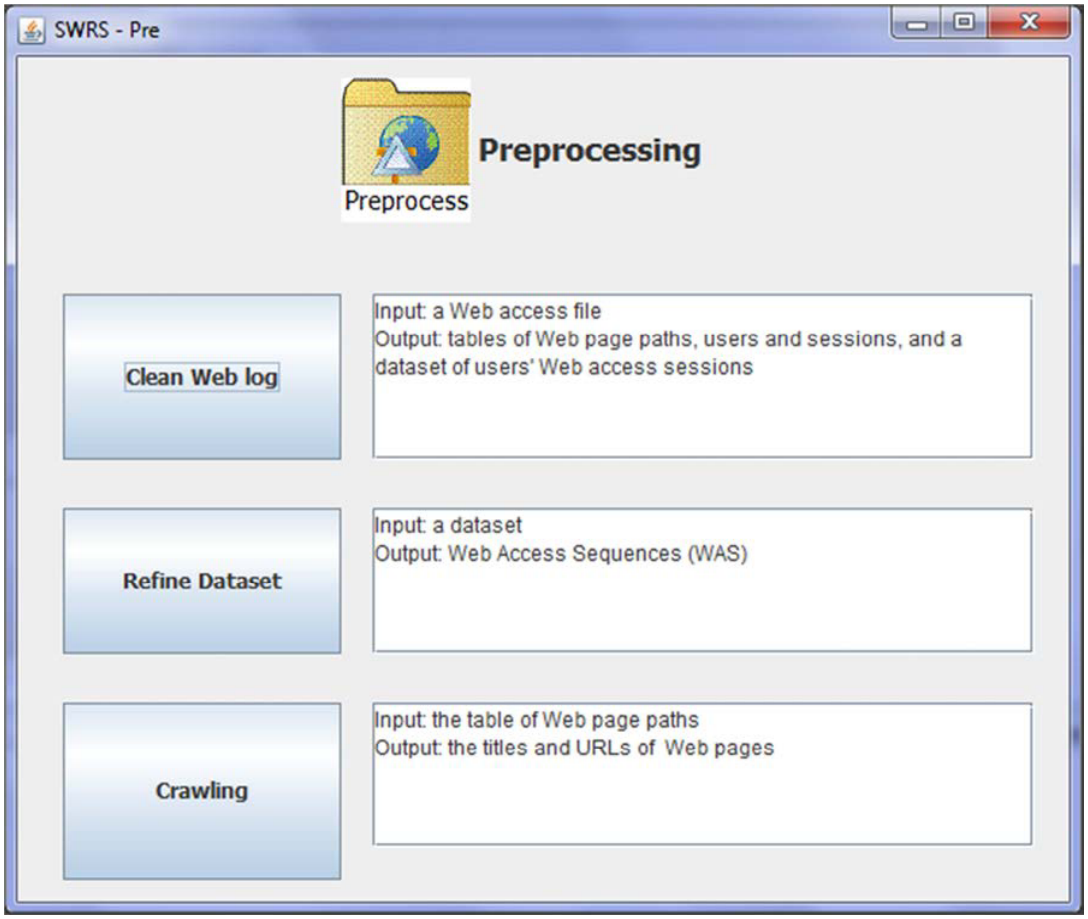


Figure 8-15: Pre-processing frame

(2) Web Usage Mining

The Web usage mining interface allows us to run the PLWAP-Mine algorithm with the input minimum support and a source file of WAS, as illustrated in Figure 8-16. Consequently, FWAP is recorded in a text file.

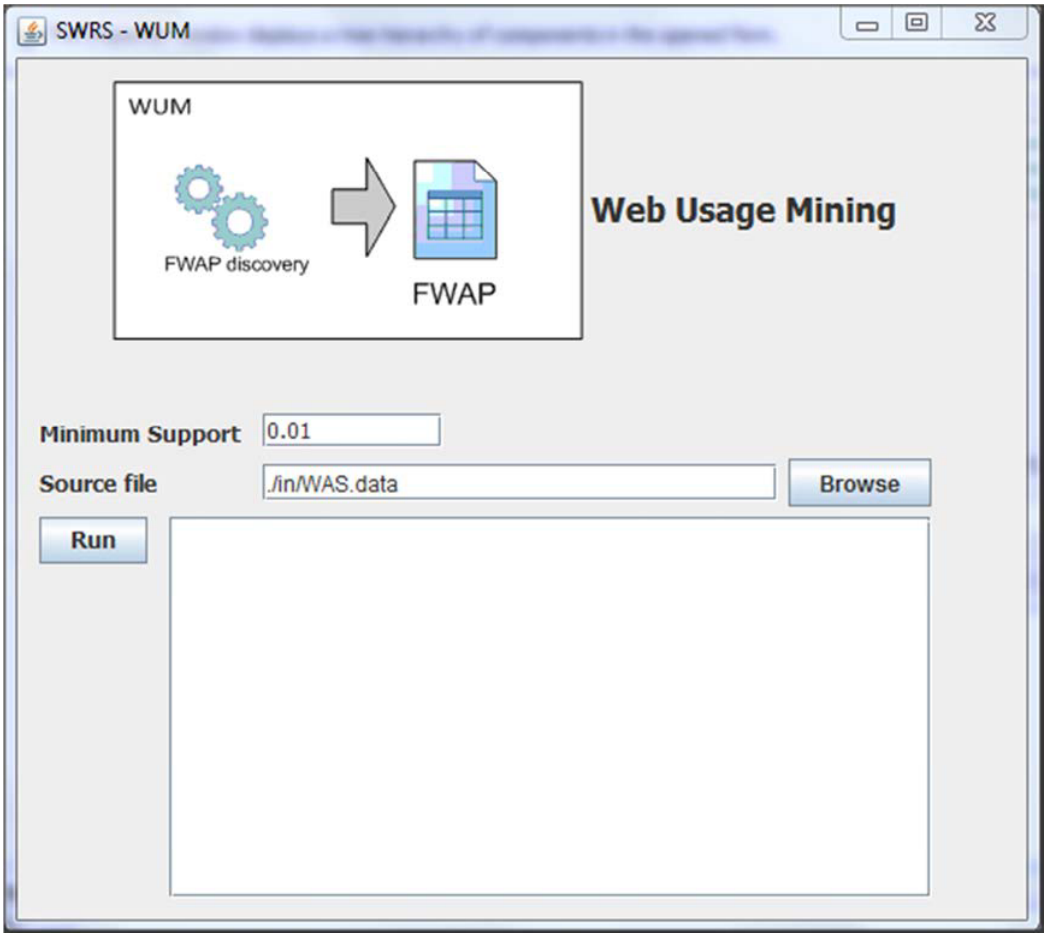


Figure 8-16: Web usage mining frame

(3) Semantic Network Construction

The semantic network construction interface allows us to generate an OWL file of TermNetWP by inputting a source file containing Web-page titles and a source label, which is used to name Page IDs, as illustrated in Figure 8-17.

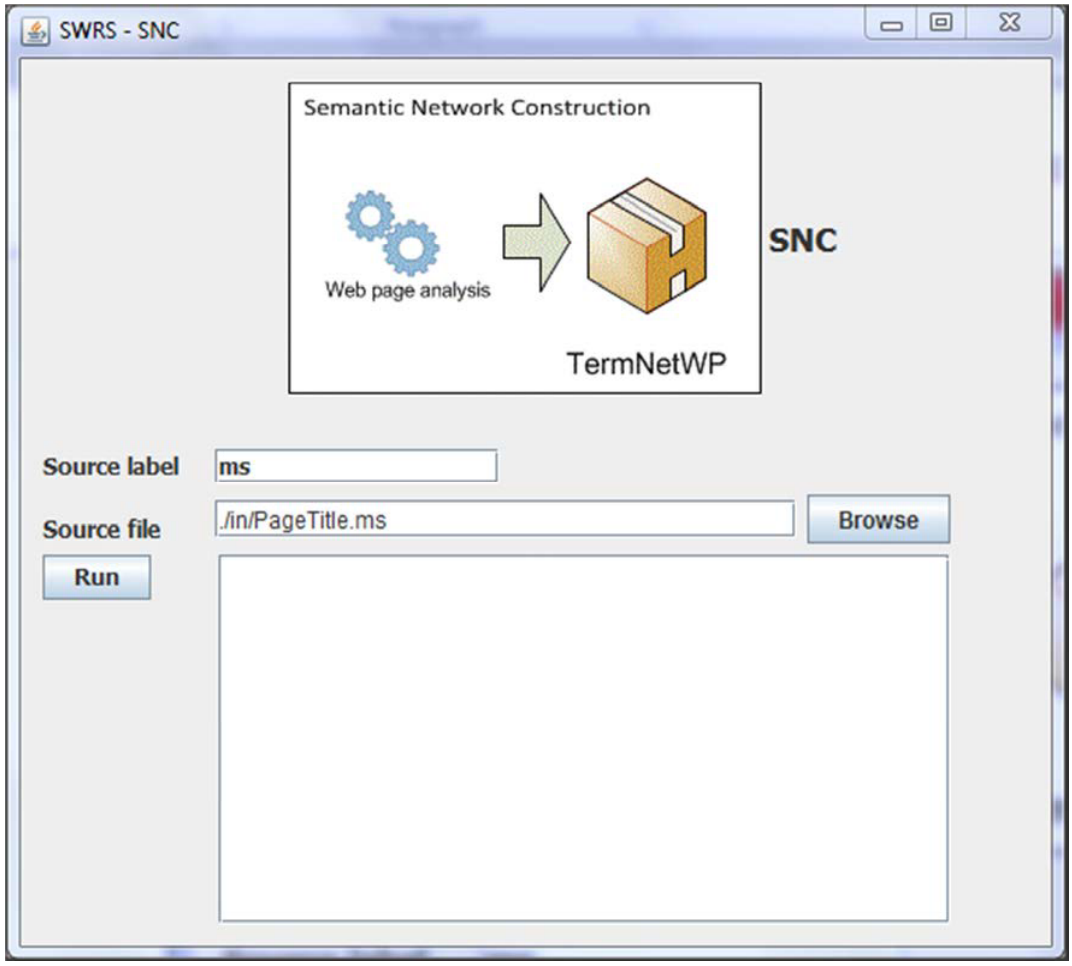


Figure 8-17: Semantic network construction frame

(4) Conceptual Prediction Model

The conceptual prediction model interface allows us to generate an OWL file of TermNavNet by inputting the PLWAP file along with the respective MinSup. The MinSup must be same as the one used to create the PLWAP file. Source label should be input as same as the one used to generate the TermNetWP. We can select the order of prediction model which is first-order or second-order. This function is illustrated in Figure 8-18.

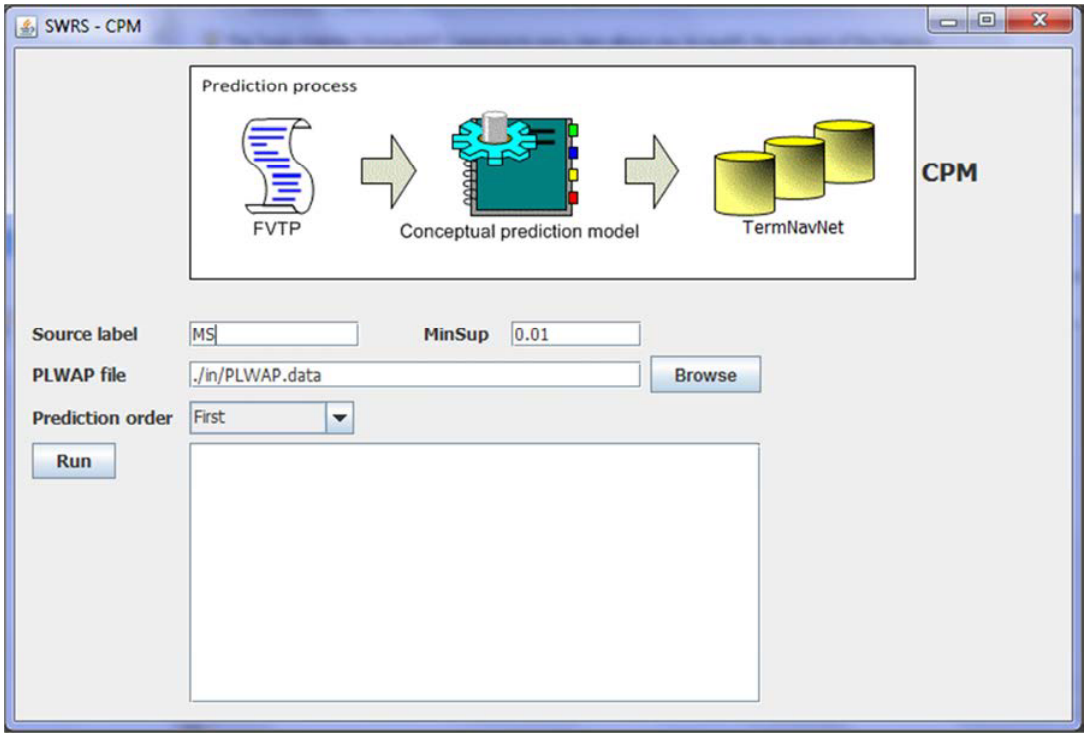


Figure 8-18: Conceptual prediction model frame

(5) Recommendation Engine

The Web-page recommendation step is performed after the above processing steps. The recommendation engine interface allows us to test the performance of the Web-page recommendation strategies by inputting the source label, the minimum support, the prediction order, a maximum recommendation length, the FWAP file, a testing file of WAS, and a selected recommendation mode which is a Web-page recommendation strategy. The source label, the minimum support, and the FWAP file must be same as the ones in the previous steps. Figure 8-19 illustrates the functions in the Recommendation engine sub-system. Button "Test" is used for testing the performance. Button "Create Freq. Pages" is used to create a set of most frequently visited Web-pages which is saved into a HTML file to be shown later in the Web browser. Button "Web UI" is used to display the Web browser which will be presented later on.

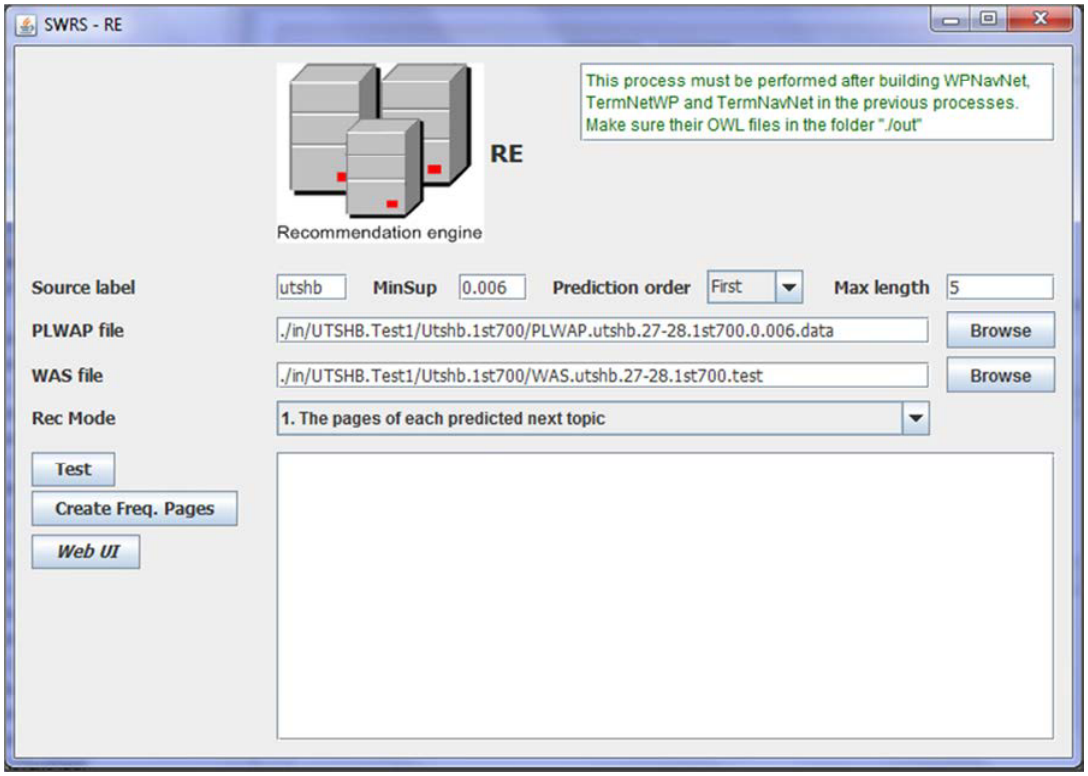


Figure 8-19: Recommendation engine frame

8.6.2. Front-end

The Web browser is used on the front-end side. It shows the most frequently visited Web-pages on the left hand side, the main page displaying the currently visited page at the center, and the recommended next pages on the right hand side, as illustrated in Figure 8-20. When we click on any link on the browser, the corresponding recommended next pages will be listed.

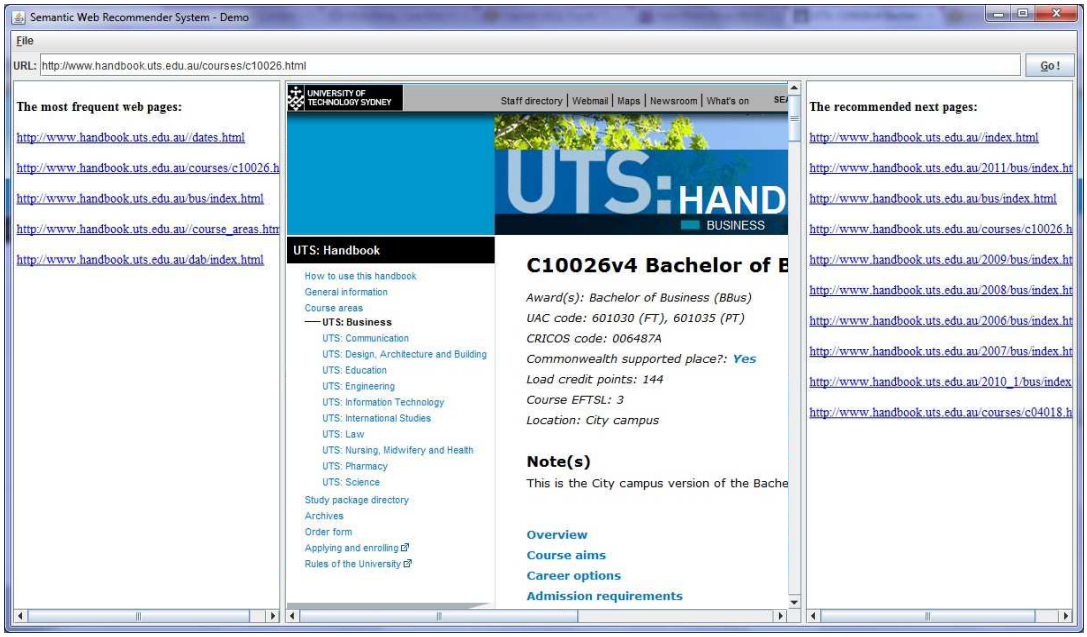


Figure 8-20: Web browser frame

8.7. Summary

In this chapter, the semantic-enhanced Web-page recommender system has been developed quite completely with the core knowledge bases represented by the proposed models, i.e., TermNetWP, DomainOntoWP, TermNavNet, and WPNavNet. The knowledge bases are represented consistently in the ontology language, so that they are machine-understandable and processed automatically. The automatic models, i.e. TermNetWP, TermNavNet, and WPNavNet, are significant for applying the system to any websites, or probably more than one websites in the same domain of interest. This enables to reduce significantly the labour-intensive tasks of building ontologies, such as domain ontology of a website.

To clarify how data is processed through the system, the main operations in the system have been presented. The user interfaces on the back-end and front-end sides have been developed to facilitate using the system. Moreover, the system allows us to test the different Web-page recommendation strategies and choose an appropriate one for given Web usage data.

Although the issue of multi-site is not highlighted in this chapter, it opens a promising issue in the future. The advantage of the system is that it is a component-based architecture, so each component can be upgrades separately. For example, we can use a different Web usage mining technique to discover FWAP, or can use a certain method of constructing a domain ontology which expresses the meanings of Web-pages. Moreover, the data structures used in the system have been specified, that would help to develop and extend the system in the future.

Chapter 9.

CONCLUSIONS AND FUTURE RESEARCH

This chapter summarizes the achievements of this study and lists possible future research directions.

9.1. Conclusions

This study aims to address the challenges and/or problems in developing Web-page recommender systems, as identified in Chapter 1, such as the "new page" problem, the challenges of manual knowledge construction, and the "heterogeneous knowledge bases" problem. This study has developed a framework for a semantic-enhanced Web-page recommender system and a suite of enabling techniques, based on which an effective Web-page recommender system can be realised. This framework makes use of the domain knowledge and Web usage knowledge of a specific website, supported by two new domain knowledge representation models and two new Web usage knowledge representation models. It also utilises a set of new recommendation strategies to offer more effective Web-page recommendations based on the integrated domain and Web usage knowledge.

The study has made significant contributions from both theoretical and practical aspects in the area of Web-page recommender systems, as detailed below:

From theoretical aspect, this study has developed a conceptual framework to facilitate the discovery, representation and integration of the useful knowledge of a website, including the domain and Web usage knowledge, to support effective Web-page recommendations.

The Web usage knowledge is the frequent Web access patterns (FWAP) of website users, which is discovered from Web logs using an advanced sequence mining technique, namely PLWAP-Mine.

The Web usage knowledge is then transformed into a weighted network of Web-pages, namely WPNavNet. Each node in the network represents a Web-page and each edge represents the transition from one Web-page to another; the weight of each edge represents the transition probability.

The domain knowledge is the knowledge about the titles of Web-pages within the specific website. The domain knowledge can be represented by either a domain ontology model, namely DomainOntoWP, or a semantic network model, namely TermNetWP. With the domain ontology model (DomainOntoWP), domain terms, the relationships between the terms and axioms are determined by domain experts, while the links between domain terms and Web-pages are automatically established by a smart software algorithm, namely the keyword-based mapping algorithm. With the semantic network model (TermNetWP), the domain terms and the associations between these terms are automatically extracted from the titles of Web-pages from the given website, and the semantic network of Web-pages can be automatically built up for the given website.

The domain knowledge represented by either DomainOntoWP or TermNetWP is then used to interpret Web-pages in FWAP by the mapping between Web-pages and domain terms to generate frequently viewed domain term patterns (FVTP). Given the FVTP, the domain term navigation model has been proposed to automatically build a weighted network of domain term navigation, namely TermNavNet.

By means of this suite of techniques, i.e., DomainOntoWP or TermNetWP as the representation of domain knowledge, TermNavNet as the semantic-enhanced Web usage knowledge, and WPNavNet as the Web usage knowledge, a set of Web-page recommendation strategies has been proposed to offer more effective Web-page recommendations. For a given user, the system can predict the Web-pages most likely to be sought based on his or her last visited Web-page(s) by using WPNavNet or TermNavNet. Using WPNavNet can quickly provide the predicted Web-pages. However, in the event that the last visited page(s) is not in WPNavNet, TermNavNet is useful for Web-page prediction. Moreover, TermNavNet-based prediction can enrich the pool of the predicted Web-pages. To use TermNavNet for Web-page prediction, the system needs to (i) extract

the domain terms of the input Web-pages based on the domain knowledge base, (ii) predict the next most likely viewed domain terms based on TermNavNet, and (iii) map the predicted domain terms back to the Web-pages, using the domain knowledge base to offer Web-page recommendations.

From the practical application aspect, this study has developed a prototype of semantic-enhanced Web-page recommender systems, in which the proposed knowledge representation models are efficiently coordinated to support effective Web-page recommendations. The experimental study has shown that the framework and the enabling techniques are valid.

In the prototype, the domain and Web usage knowledge bases are implemented in OWL, which is a commonly used ontology language. In this way, the various knowledge bases, DomainOntoWP, TermNetWP, TermNavNet and WPNavNet, are seamlessly integrated in an automated fashion. These knowledge bases support the set of recommendation strategies which can predict the Web-pages that are next most likely to be visited for a given user. A number of experiments have been conducted to compare the effectiveness of the knowledge representation models and recommendation strategies.

Three types of Web-page recommendation strategies have been implemented: Type I, which has no semantic enhancement, i.e. it is solely based on WPNavNet or PLWAP-Mine; Type II, which has semantic enhancement based on the domain ontology, i.e. coordinates WPNavNet and the DomainOntoWP-based TermNavNet; and Type III, which has semantic enhancement based on the semantic network, i.e. coordinates WPNavNet and the TermNetWP-based TermNavNet. The test datasets used are from real world websites; one is publicly available from the UCI machine learning repository, and one is from the University of Technology, Sydney (UTS). The experimental evaluation metrics are precision and satisfaction.

The experimental comparisons show that types II and III achieve higher performance than Type I, thanks to the semantic enhancement. As remarked in Section 7.4.3, Type III always outperforms Type II, so is the best solution for Web-page recommendation in websites. In particular, the first-order navigation models, i.e. the first-order WPNavNet, and

the first-order TermNavNet, are more effective than the second-order navigation models in event prediction. In Type III, based on the same order navigation models, considering the prediction probabilities in TermNavNet, and taking into account all sets of the predicted domain terms or each set of next domain terms, in which each set of next domain terms is predicted given the current domain term(s) based on TermNavNet, the Web-page recommendation achieves the highest performance.

Generally speaking, this study has addressed the major problems and challenges in Web-page recommendation systems. For the "new page" problem, this study interprets the semantics of the new page to generate recommendations with the aid of the domain knowledge bases. If the domain terms of the new page are in the domain knowledge bases, which is true in most cases, the system can generate recommendations to users. If the domain terms of the new page are not in the knowledge bases, the system needs to update the knowledge bases by adding the new page, new terms and relationships to the knowledge bases. In terms of knowledge base update, the semantic network model is more convenient than the domain ontology model because it can be run automatically at any time, whereas the domain ontology model might need a developer's support. With the updated knowledge bases, a user visiting a new page can obtain useful Web-page recommendations. Therefore, this study has significantly alleviated the "new page" problem.

To tackle the challenges in manual domain ontology construction, this study developed a smart algorithm to map Web-pages to respective domain terms in the domain ontology model, and developed a smart algorithm to build the semantic network of Web-pages for the semantic network model. This development contributes significantly to domain knowledge discovery and representation, and has a great extension to automatic domain knowledge construction.

For the "heterogeneous knowledge bases" problem, this study developed a set of techniques to represent the various knowledge models in ontological style by taking advantage of the expressive power of ontology language. This enables the automation of

processes in the integration of knowledge bases, and seamless coordination in generating Web-page recommendations.

In conclusion, this study has achieved its objectives and has proved to be very successful. The theoretical development and practical advancements creates a great opportunity to develop new generation intelligent Web-page recommender systems.

9.2. Future Research

Given the time constraints of this study, the scope of the system development was limited to (i) applying to a single website, (ii) discovering static Web-pages, (iii) extracting the semantics of Web-pages from the pages' titles, and (iv) using a segment of Web log. In future research, the following four directions can be considered:

(i) Multi-site

The proposed framework and enabling techniques can be extended to make Web-page recommendations for multiple websites in the same domain. Along this direction, the system should take the log files from these websites as the input. Although the current system can be applied to multiple sites, a deep look at the semantic analysis of domain terms in different sites is necessary to achieve effective recommendation results. The main work would take the form of knowledge base construction, with the focus on handling similar domain terms in building the semantic network, i.e. TermNetWP. Similar domain terms will be clustered or generalized to form different concept levels in the semantic network. Therefore, the system can generate Web-page recommendations based not only on the domain terms of Web-pages, but also on the similarity of the domain terms between websites. For example, there is a website of courses, e.g. UTS handbook, and a website of textbooks, e.g. wikibooks. The Web logs of both websites can be input into the system. When a user visits one or more courses, the system might not only recommend pages of relevant courses but also pages of relevant books.

(ii) Web usage data

The current system works with static Web-pages. With the advancement in Web technology, pages have been evolving into pages with dynamic structures. To offer more effective Web-page recommendations, it will be highly desirable to develop advanced tools to identify and collect more appropriate Web usage data than Web logs, such as clickstream data. Dynamic Web clickstream analysis can be conducted in the data preparation stage, in which the Web-pages may be identified as dynamic contents or "items", rather than static pages. In addition, in large websites, e.g. university websites, there are various areas in relation to courses, student services, and different types of staff. Web usage data will need to be clustered in specific domains on the website, such as courses, student services, or staff, so that recommendations can be focused on more relevant information. The system can be applied to a specific area on the website, and a clustering method might be applied to select Web-pages that focus on a specific domain in the pre-processing stage.

(iii) Domain knowledge discovery and representation

Advanced topic models from the area of information retrieval can be used to extract the domain terms from the Web-pages on the website. Synonyms of the domain terms may need to be identified using WordNet, and domain terms may need to be clustered into common topics. This would help to cluster Web-pages into relevant topics and thereby optimize the semantic network model of Web-pages. In addition, the domain term navigation modelling in the current system takes an intermediate process step to generate TermNavNet from FWAP. This means that a set of FVTP needs to be generated for given FWAP and then transformed to a TermNavNet. Since the used domain term patterns, i.e. FVTP, are the result of the integration of FWAP and the domain terms of Web-pages, a large number of domain terms from each page might lead to a large number of domain term patterns being generated. Therefore, it will be necessary to limit the number of domain terms for each page, and to select proper domain terms in the construction of TermNavNet to save memory space and to reduce the running time of building and reasoning algorithms.

(iv) Web usage knowledge base update

Websites have been evolving over time therefore the knowledge bases, including the domain and Web usage knowledge bases, need to be updated accordingly. Considering the traditional Web usage data source, which is the Web log file, the system can only take a limited segment of the log file to build the Web usage knowledge base due to the fact that the size of the log file can be huge. To ensure that the discovered Web usage knowledge is up-to date, new methods need to be developed to dynamically update the knowledge bases. For example, an incremental mining method, e.g. PLWAP For UPdate (PL4UP) (Ezeife & Liu 2009), can be utilised to update FWAP, which is discovered from the Web usage data. Updating the WPNavNet and TermNavNet models will be taken into account; for example, the generated WPNavNet can be updated when there are changes to FWAP.

ABBREVIATIONS

Abbreviations	Descriptions
CF	Collaborative Filtering
CID	Class Instance Distribution
CLF	Common Log Format
CLP	Concept Location Precision
CP	Concept Precision
CR	Class Richness
CS-Mine	Conditional Sequence Mining
DOC	Domain Ontology Construction
DomainOntoWP	Domain Ontology of Web-pages
EM	Expectation–Maximization
FP	Frequent Patterns
FVTP	Frequently Viewed Topic Patterns
FWAP	Frequent Web Access Patterns
HTML	HyperText Markup Language
GRSK	Generalist Recommender System Kernel
GSP	Generalized Sequential Pattern
LDA	Latent Dirichlet Allocation
OWL	Web Ontology Language
PLWAP-Mine	Pre-order Linked Web Access Pattern tree Mining
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RDQL	RDF Data Query Language
RE	Recommendation Engine
RIF	Rule Interchange Format
RR	Relationship Richness
RU	Relationship Utilization
SNC	Semantic Network Construction
SPADE	Sequential PAttern Discovery using Equivalent Class
SQL	Structured Query Language
SWRS	Semantic-enhanced Web-page Recommender System
TermNetWP	Semantic Network of Web-pages
TF	Term Frequency
TF-IDF	Term Frequency–Inverse Document Frequency
TLP	Term Location Precision
TermNavNet	Topic Navigation Network
TP	Term Precision
URL	Uniform Resource Locator
WAP	Web Access Patterns
WAS	Web Access Sequences
WCM	Web Content Mining
WM	Web Mining
WPNavNet	Web-page Navigation Network
WSM	Web Structure Mining
WUM	Web Usage Mining
XML	Extensible Markup Language

BIBLIOGRAPHY

Adomavicius, G. & Tuzhilin, A. 2005, 'Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions', *IEEE Trans. on Knowl. and Data Eng.*, vol. 17, no. 6, pp. 734-749.

Agrawal, R. & Srikant, R. 1995, 'Mining Sequential Patterns', *Proceedings of the Eleventh International Conference on Data Engineering*, IEEE Computer Society, Taipei, pp. 3-14.

Alani, H., Kim, S., Millard, D., Weal, M., Hall, W., Lewis, P. & Shadbolt, N. 2003, 'Automatic Ontology-Based Knowledge Extraction from Web Documents', *IEEE Intelligent Systems*, vol. 18, no. 1, pp. 14-21.

Alves, A., Antunes, B., Pereira, F.C. & Bento, C. 2009, 'Semantic Enrichment of Places: Ontology Learning from Web', *International Journal of Knowledge-Based & Intelligent Engineering Systems*, vol. 13, no. 1, pp. 19-30.

Antoniou, G. & Harmelen, F.v. 2008, *A Semantic Web Primer*, MIT Press.

Antoniou, G. & Harmelen, F.V. 2009, 'Web Ontology Language: OWL', in S. Staab & R. Studer (eds), *Handbook on Ontologies*, Springer-Verlag Berlin Heidelberg, pp. 91-110.

Berendt, B., Hotho, A. & Stumme, G. 2002, 'Towards Semantic Web Mining', in, *The Semantic Web — ISWC 2002*, pp. 264-278.

Borges, J. & Levene, M. 2004, *A Dynamic Clustering-Based Markov Model for Web Usage Mining*, Available online at http://xxx.arxiv.org/abs/cs.IR/0406032.

Borges, J. & Levene, M. 2005, 'Generating Dynamic Higher-Order Markov Models in Web Usage Mining', in, *Knowledge Discovery in Databases: PKDD 2005*, vol. 3721, Springer Berlin / Heidelberg, pp. 34-45.

Bose, A., Beemanapalli, K., Srivastava, J. & Sahar, S. 2007, 'Incorporating Concept Hierarchies into Usage Mining Based Recommendations', paper presented to the *Proceedings of the 8th Knowledge discovery on the web international conference on Advances in web mining and web usage analysis*, Philadelphia, PA, USA.

Boyce, S. & Pahl, C. 2007, 'Developing Domain Ontologies for Course Content', *Educational Technology & Society*, vol. 10, no. 3, pp. 275-288.

Bürger, T. & Simperl, E. 2010, 'Measuring the Benefits of Ontologies', in, *On the Move to Meaningful Internet Systems: OTM 2008 Workshops*, vol. 584-594.

Cimiano, P., M¨adche, A., Staab, S. & V¨olker, J. 2009, 'Ontology Learning', in S. Staab & R. Studer (eds), *Handbook on Ontologies*, Springer-Verlag Berlin Heidelberg, pp. 245 - 267.

Chen, L., Bhowmick, S.S. & Li, J. 2006, 'COWES: Clustering Web Users Based on Historical Web Sessions', in, *Database Systems for Advanced Applications*, vol. 3882/2006, Springer Berlin / Heidelberg, pp. 541-556.

Chen, R.-C. & Chuang, C.-H. 2008, 'Automating Construction of a Domain Ontology using a Projective Adaptive Resonance Theory Neural Network and Bayesian Network', *Expert Systems*, vol. 25, no. 4, pp. 414-430.

Dai, H. & Mobasher, B. 2005, 'Integrating Semantic Knowledge with Web Usage Mining for Personalization', in A. Scime (ed.), *Web Mining: Applications and Techniques*, IGI Global, Hershey, PA, USA, pp. 276 - 306.

Dai, H., Srikant, R., Zhang, C., Wu, S. & Wang, W. 2004, 'Predicting Web Requests Efficiently Using a Probability Model', in, *Advances in Knowledge Discovery and Data Mining*, vol. 3056, Springer Berlin / Heidelberg, pp. 564-568.

Deshpande, M. & Karypis, G. 2004, 'Selective Markov Models for Predicting Web-Page Accesses', *ACM Transactions on Internet Technology (TOIT)*, vol. 4, no. 2, pp. 163 - 184.

Di Martino, B. & Cantiello, P. 2009, 'Automatic Ontology Extraction with Text Clustering', in G. Papadopoulos & C. Badica (eds), *Intelligent Distributed Computing III*, Springer Berlin / Heidelberg, pp. 215 - 220.

Dixit, D. & Gadge, J. 2010, 'Automatic Recommendation for Online Users Using Web Usage Mining', *IJMIT*, vol. 2, no. 3, pp. 33-42.

Domingue, J., Fensel, D. & Hendler, J.A. 2011, 'Introduction to the Semantic Web Technologies', in J. Domingue, D. Fensel & J.A. Hendler (eds), *Handbook of Semantic Web Technologies*, Springer-Verlag Berlin Heidelberg, pp. 3-41.

Drury, B. & Almeida, J. 2009, 'Construction of a Local Domain Ontology from News Stories', in, *Progress in Artificial Intelligence*, vol. 5816, Springer Berlin / Heidelberg, pp. 400-410.

Dzemydiene, D. & Tankeleviciene, L. 2008, 'On the Development of Domain Ontology for Distance Learning Course', paper presented to the *20th EURO Mini Conference "Continuous Optimization and Knowledge-Based Technologies"*, Neringa, LITHUANIA.

Eirinaki, M., Mavroeidis, D., Tsatsaronis, G. & Vazirgiannis, M. 2006, 'Introducing Semantics in Web Personalization: The Role of Ontologies', in M.A.e. al. (ed.), *Semantics, Web, and Mining*, Springer-Verlag Berlin Heidelberg, pp. 147 – 162.

Eirinaki, M. & Vazirgiannis, M. 2007, 'Web Site Personalization Based on Link Analysis and Navigational Patterns', *ACM Transactions on Internet Technology*, vol. 7, no. 4, p. 27.

Ezeife, C. & Liu, Y. 2009, 'Fast Incremental Mining of Web Sequential Patterns with PLWAP Tree', *Data Mining and Knowledge Discovery*, vol. 19, no. 3, pp. 376-416.

Ezeife, C.I. & Lu, Y. 2005, 'Mining Web Log Sequential Patterns with Position Coded Pre-Order Linked WAP-Tree', *Data Mining and Knowledge Discovery*, vol. 10, no. 1, pp. 5-38.

Garcia, I., Sebastia, L., Pajares, S. & Onaindia, E. 2010, 'GRSK: A Generalist Recommender System', *WEBIST (1)*, Spain, pp. 211-218.

Gascuena, J.M., Fernandez-Caballero, A. & Gonzalez, P. 2006, 'Domain Ontology for Personalized E-Learning in Educational Systems', *Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies*, IEEE Computer Society, pp. 456 - 458.

Grimm, S., Abecker, A., Völker, J. & Studer, R. 2011, 'Ontologies and the Semantic Web', in J. Domingue, D. Fensel & J.A. Hendler (eds), *Handbook of Semantic Web Technologies*, Springer-Verlag Berlin Heidelberg, pp. 507-580.

Grinstead, C.M. & Snell, J.L. 1997, *Introduction to Probability*, American Mathematical Society.

Gruber, T.R. 1995, 'Toward Principles for the Design of Ontologies used for Knowledge Sharing', *Int. J. Hum.-Comput. Stud.*, vol. 43, no. 5-6, pp. 907-928.

Guarino, N., Oberle, D. & Staab, S. 2009, 'What is an Ontology?', in S. Staab & R. Studer (eds), *Handbook on Ontologies*, vol. 2, Springer-Verlag Berlin Heidelberg, pp. 1-17.

Gündüz-Ögüdücü, Ş. 2010, *Web Page Recommendation Models: Theory and Algorithms*, Morgan & Claypool.

Han, J., Pei, J., Asl, B., Chen, Q., Dayal, U. & Hsu, M. 2000, 'FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining', *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, Boston, Massachusetts, United States, pp. 355-359.

Han, J., Pei, J. & Yan, X. 2005, 'Sequential Pattern Mining by Pattern-Growth: Principles and Extensions', in, *Foundations and Advances in Data Mining*, vol. 180, Springer Berlin / Heidelberg, pp. 183-220.

Han, J., Pei, J. & Yin, Y. 2000, 'Mining Frequent Patterns without Candidate Generation', *SIGMOD Rec.*, vol. 29, no. 2, pp. 1-12.

Harth, A., Janik, M. & Staab, S. 2011, 'Semantic Web Architecture', in J. Domingue, D. Fensel & J.A. Hendler (eds), *Handbook of Semantic Web Technologies*, Springer-Verlag Berlin Heidelberg, pp. 43-75.

Henze, N., Dolog, P. & Nejdl, W. 2004, 'Reasoning and Ontologies for Personalized E-Learning in the Semantic Web', *Educational Technology & Society*, vol. 7, no. 4, pp. 82-97.

Hepp, M. 2008, 'Ontologies: State of the Art, Business Potential, and Grand Challenges', in M. Hepp, P.D. Leenheer, A.d. Moor & Y. Sure (eds), *Ontology Management: Semantic Web, Semantic Web Services, and Business Applications*, Springer, Heidelberg, pp. 3-22.

Horrocks, I. & Peter, F.P.-S. 2011, 'KR and Reasoning on the Semantic Web: OWL', in J. Domingue, D. Fensel & J.A. Hendler (eds), *Handbook of Semantic Web Technologies*, Springer-Verlag Berlin Heidelberg, pp. 365-398.

Hu, H., Du, X.Y., Liu, D.Y. & Ouyang, J.H. 2006, 'Ontology Learning using WordNet Lexicon', in G.R. LIU, V.B.C. TAN & X. HAN (eds), *Computational Methods*, Springer Netherlands, pp. 1249-1253.

Huffman, D. 1952, 'A Method for the Construction of Minimum-Redundancy Codes', *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098-1101.

Jalali, M., Mustapha, N., Sulaiman, M.N. & Mamat, A. 2009, 'A Recommender System Approach for Classifying User Navigation Patterns Using Longest Common Subsequence Algorithm', *American Journal of Scientific Research*, vol. 10, no. 4, pp. 17-27.

Kolari, P. & Joshi, A. 2004, 'Web Mining: Research and Practice', *Computing in Science and Engineering*, vol. 6, pp. 49-53.

Konstan, J. & Riedl, J. 2012, 'Recommender Systems: from Algorithms to User Experience', *User Modeling and User-Adapted Interaction*, vol. 22, no. 1, pp. 101-123.

Khalil, F. 2008, 'Combining Web Data Mining Techniques for Web Page Access Prediction', Doctoral thesis thesis, University of Southern Queensland.

Khribi, M.K.r., Jemni, M. & Nasraoui, O. 2009, 'Automatic Recommendations for E-Learning Personalization Based on Web Usage Mining Techniques and Information Retrieval', *Journal of Educational Technology & Society*, vol. 12, no. 4, pp. 30-42.

Lam, S. & Riedl, J. 2004, 'Shilling Recommender Systems for Fun and Profit', *the 13th international conference on World Wide Web*, pp. 393-402.

Li, L. & Zhang, W. 2008, 'Higher-Order Logic Recommender System', paper presented to the *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*.

Li, Y. & Zhong, N. 2003, 'Ontology-based Web mining model: representations of user profiles', *Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI 2003)*, pp. 96-103.

Li, Y. & Zhong, N. 2004, 'Capturing Evolving Patterns for Ontology-based Web Mining', paper presented to the *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*.

Li, Y. & Zhong, N. 2006, 'Mining Ontology for Automatically Acquiring Web User Information Needs', *IEEE Transactions on Knowledge & Data Engineering*, vol. 18, no. 4, pp. 554-568.

Liang, W. & Zhao, S.-h. 2009, 'A Hybrid Recommender System Combining Web Page Clustering with Web Usage Mining', *International Conference on Computational Intelligence and Software Engineering, 2009.* , Wuhan, pp. 1-4.

Liu, B. 2011a, 'Information Retrieval and Web Search', in B. Liu (ed.), *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, Springer-Verlag Berlin Heidelberg, pp. 183-236.

Liu, B. (ed.) 2011b, *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data* Springer-Verlag Berlin Heidelberg.

Liu, B., Mobasher, B. & Nasraoui, O. 2011, 'Web Usage Mining', in B. Liu (ed.), *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, Springer-Verlag Berlin Heidelberg, pp. 527-603.

Liu, Y., Huang, X. & An, A. 2007, 'Personalized Recommendation with Adaptive Mixture of Markov Models', *The American Society for Information Science and Technology*, vol. 58, no. 12, pp. 1851–1870.

Loizou, A. & Dasmahapatra, S. 2006, 'Recommender Systems for the Semantic Web', paper presented to the *ECAI 2006 Recommender Systems Workshop*, Trento, Italy.

Lovrenčić, S. & Čubrilo, M. 2008, 'Ontology Evaluation – Comprising Verification and Validation', *Proceedings of the 19th Central European Conference on Information and Intelligent Systems*, pp. 657-663.

Lu, J., Wang, C., Zhang, G. & Ma, J. 2009, 'Collaborative Management of Web Ontology Data with Flexible Access Control', *Expert Syst. Appl.*, vol. 37, no. 5, pp. 3737-3746.

Mabroukeh, N.R. & Ezeife, C.I. 2009, 'Semantic-Rich Markov Models for Web Prefetching', paper presented to the *2009 IEEE International Conference on Data Mining Workshops*, Miami, Florida, USA.

Mabroukeh, N.R. & Ezeife, C.I. 2010, 'A Taxonomy of Sequential Pattern Mining Algorithms', *ACM Comput. Surv.*, vol. 43, no. 1, pp. 1-41.

Maedche, A.D. 2002, *Ontology Learning for the Semantic Web*, Kluwer Academic Publishers.

Markov, Z. & Larose, D.T. 2007a, 'Chapter 1: Information Retrieval and Web Search', in Z. Markov & D.T. Larose (eds), *Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage*, John Wiley & Sons, Inc, New Britain, pp. 3-46.

Markov, Z. & Larose, D.T. 2007b, *Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage*, John Wiley & Sons, Inc, New Britain.

Menczer, F. 2011, 'Web Crawling', in B. Liu (ed.), *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, Springer-Verlag Berlin Heidelberg, pp. 273-321.

Mobasher, B. 2007a, 'Data Mining for Web Personalization', in P. Brusilovsky, A. Kobsa & W. Nejdl (eds), *The Adaptive Web*, vol. 4321, Springer-Verlag Berlin, Heidelberg, pp. 90-135.

Mobasher, B. 2007b, *Recommender Systems*.

Mobasher, B., Burke, R., Bhaumik, R. & Williams, C. 2007, 'Toward Trustworthy Recommender Systems: An Analysis of Attack Models and Algorithm Robustness', *ACM Transactions on Internet Technology*, vol. 7, no. 4, p. 23.

Mobasher, B., Dai, H., Luo, T. & Nakagawa, M. 2002, 'Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization', *Data Min. Knowl. Discov.*, vol. 6, no. 1, pp. 61-82.

Nasraoui, O., Soliman, M., Saka, E., Badia, A. & Germain, R. 2008, 'A Web Usage Mining Framework for Mining Evolving User Profiles in Dynamic Web Sites', *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, pp. 202-215.

Oberle, D., Grimm, S. & Staab, S. 2009, 'An Ontology for Software', in S. Staab & R. Studer (eds), *Handbook on Ontologies*, vol. 2, Springer, Berlin, Heidelberg, pp. 383-402.

Park, Y.C., Han, Y.S. & Choi, K.-S. 1995, 'Automatic Thesaurus Construction using Bayesian Networks', paper presented to the *Proceedings of the fourth international conference on Information and knowledge management*, Baltimore, Maryland, United States.

Pei, J., Han, J., Asl, M., Pinto, H., Chen, Q., Dayal, U. & Hsu, M.C. 2001, 'PrefixSpan Mining Sequential Patterns Efficiently by Prefix Projected Pattern Growth', *Proc.17th Intl Conf. on Data Eng.*, pp. 215-226.

Pei, J., Han, J., Mortazavi-asl, B. & Zhu, H. 2000, 'Mining Access Patterns Efficiently from Web Logs', in, *Knowledge Discovery and Data Mining. Current Issues and New Applications*, vol. 1805, Springer Berlin / Heidelberg, pp. 396-407.

Pierrakakos, D., Paliouras, G., Papatheodorou, C. & Spyropoulos, C.D. 2003, 'Web Usage Mining as a Tool for Personalization: A Survey', *User Modelling and User-Adapted Interaction*, vol. 13, no. 4, pp. 311-372.

Ricci, F., Rokach, L. & Shapira, B. 2011, 'Introduction to Recommender Systems Handbook', in F. Ricci, L. Rokach, B. Shapira & P.B. Kantor (eds), *Recommender Systems Handbook*, Springer Science+Business Media, pp. 1-35.

Rios, S.A. & Velasquez, J.D. 2008, 'Semantic Web Usage Mining by a Concept-Based Approach for Off-line Web Site Enhancements', *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT '08. IEEE/WIC/ACM International Conference on*, vol. 1, pp. 234-241.

Ruiz-Montiel, M. & Aldana-Montes, J.F. 2009, 'Semantically Enhanced Recommender Systems', in R. Meersman, P. Herrero & T. Dillon (eds), *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, Springer-Verlag Berlin Heidelberg, pp. 604-609.

Salin, S. & Senkul, P. 2009, 'Using Semantic Information for Web Usage Mining based Recommendation', *24th International Symposium on Computer and Information Sciences, 2009.*, pp. 236-241.

Sánchez, D. & Moreno, A. 2006, 'Discovering Non-taxonomic Relations from the Web', *7th International Conference on Intelligent Data Engineering and Automated Learning*, pp. 629-636.

Schutz, A. & Buitelaar, P. 2005, 'RelExt: A Tool for Relation Extraction from Text in Ontology Extension', *ISWC 2005*, vol. 3729, Springer, Galway, Ireland, pp. 593–606.

Semeraro, G., Degemmis, M., Lops, P., Basile, P. & Gentile, A.L. 2007, 'Semantic Web Personalization in a Scientific Congress Scenario', *CEUR Workshop Proceedings*, vol. 201.

Shani, G. & Gunawardana, A. 2011, 'Evaluating Recommendation Systems', in F. Ricci, L. Rokach, B. Shapira & P.B. Kantor (eds), *Recommender Systems Handbook*, Springer Science+Business Media, pp. 257-297.

Sosnovsky, S. 2008, 'Ontological Technologies for User Modeling', PhD. thesis, University of Pittsburgh.

Sowa, J.F. 1991, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Morgan Kaufmann.

Srikant, R. & Agrawal, R. 1996, 'Mining Sequential Patterns: Generalizations And Performance Improvements', *In Proceedings of the 5th International Conference on Extending Database Technology (EDBT)*, Avignon, France, pp. 3-17.

Stumme, G., Hotho, A. & Berendt, B. 2006, 'Semantic Web Mining: State of the art and future directions', *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 4, no. 2, pp. 124-143.

Tartir, S. & Arpinar, I.B. 2007, 'Ontology Evaluation and Ranking using OntoQA', paper presented to the *Proceedings of the International Conference on Semantic Computing*.

Tartir, S., Arpinar, I.B., Moore, M., Sheth, A.P. & Aleman-meza, B. 2005, 'OntoQA: Metric-based Ontology Quality Analysis', paper presented to the *IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*

Uschold, M. & Gruninger, M. 1996, 'Ontologies: Principles, Methods and Applications', *Knowledge Engineering Review*, vol. 11, pp. 93-36.

Vrandečic, D. 2009, 'Ontology Evaluation', in S. Staab & R. Studer (eds), *Handbook on Ontologies*, vol. 2, Springer, Berlin, Heidelberg, pp. 293-313.

Wang, H., Liu, Z. & Yu, C. 2007, 'Formal Ontology Model and its Application to Semantic Retrieval', *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pp. 5621-5624.

Wang, X., Bai, Y. & Li, Y. 2010, 'An Information Retrieval Method Based on Sequential Access Patterns', *2010 Asia-Pacific Conference on Wearable Computing Systems (APWCS)*, pp. 247-250.

Wei, L. & Lei, S. 2009, 'Integrated Recommender Systems Based on Ontology and Usage Mining', in J. Liu, J. Wu, Y. Yao & T. Nishida (eds), *Active Media Technology*, vol. 5820, Springer-Verlag Berlin Heidelberg, pp. 114–125.

Woon, Y.-K., Ng, W.-K. & Lim, E.-P. 2005, 'Web Usage Mining: Algorithms and Results', in A. Scime (ed.), *Web Mining: Applications and Techniques*, IGI, pp. 373-392.

Xiaohui, T., Yuefeng, L. & Nayak, R. 2008, 'A Knowledge Retrieval Model using Ontology Mining and User Profiling', *Integrated Computer-Aided Engineering*, vol. 15, no. 4, pp. 313-329.

Xu, G., Zhang, Y. & Yi, X. 2008, 'Modelling User Behaviour for Web Recommendation Using LDA Model', paper presented to the *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*.

Yan, G., Kui, L., Kai, Z. & Gang, Z. 2006, 'Board Forum Crawling: A Web Crawling Method for Web Forum', *IEEE/WIC/ACM International Conference on Web Intelligence, 2006.*, pp. 745-748.

Zaki, M. 2001, 'SPADE: An Efficient Algorithm for Mining Frequent Sequences', *Machine Learning*, vol. 42, no. 1/2, pp. 31-60.

Zhang, Q. & Segall, R.S. 2008, 'Web Mining: A Survey of Current Research, Techniques, and Software', *International Journal of Information Technology & Decision Making*, vol. 7, no. 4, pp. 683-720.

Zhou, B. 2004, *Intelligent Web Usage Mining*, Nanyang Technological University.

Zhou, B., Hui, S.C. & Chang, K. 2004, 'An Intelligent Recommender System using Sequential Web Access Patterns', *IEEE Conference on Cybernetics and Intelligent Systems*, vol. 1, pp. 393-398.

Zhou, B., Hui, S.C. & Fong, A.C.M. 2004, 'CS-Mine: An Efficient WAP-Tree Mining for Web Access Patterns', in, *Advanced Web Technologies and Applications*, vol. 3007, Springer Berlin / Heidelberg, pp. 523-532.

Zhou, B., Hui, S.C. & Fong, A.C.M. 2005, 'Web Usage Mining for Semantic Web Personalization', *PerSWeb'05 Workshop on Personalization on the Semantic Web*, Edinburgh, UK.

Zhu, J., Hong, J. & Hughes, J.G. 2002, 'Using Markov Chains for Link Prediction in Adaptive Web Sites', paper presented to the *Proceedings of the First International Conference on Computing in an Imperfect World*.

Zhu, J., Hong, J. & Hughes, J.G. 2004, 'PageCluster: Mining Conceptual Link Hierarchies from Web Log Files for Adaptive Web Site Navigation', *ACM Transactions on Internet Technology*, vol. 4, pp. 185-208.