Electronic Theses and Dissertations   Theses, Dissertations, and Major Papers

2011

SemAware: An Ontology-Based Web Recommendation System

Nizar Mabroukeh
University of Windsor

SemAware: An Ontology-Based Web Recommendation System

by

NIZAR R. MABROUKEH

A Dissertation
Submitted to the Faculty of Graduate Studies
through Computer Science
in Partial Fulfillment of the Requirements for
The Degree of Doctor of Philosophy at the
University of Windsor

Windsor, Ontario, Canada
2011

Declaration of Previous Publication

This thesis includes four original papers that have been previously published in blind reviewed journal and conference proceedings, as follows:

Thesis Chapter	Publication title/full citation	Publication status
Part of chapter 2	Nizar R. Mabroukeh and C. I. Ezeife. A Taxonomy of Sequential Pattern Mining Algorithms. ACM Computing Surveys, 43(1), Nov. 2010.	published
Major part of chapter 5 and parts of chapters 6 and 7	Nizar R. Mabroukeh and Christie I. Ezeife. Using domain ontology for semantic web usage mining and next page prediction. In CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management, pages 1677-1680, New York, NY, USA, 2009. ACM.	published
Major part of Chapter 6	Nizar R. Mabroukeh and Christie I. Ezeife. Semantic-rich Markov Models for Web Prefetching. In IEEE International Conference on Data Mining Workshops, pages 465-470, December 6-9 2009	published
Major part of chapter 7	Nizar R. Mabroukeh and Christie I. Ezeife. Ontology-based Web Recommendation from Tags. In ICDE '11 International Conference on Data Engineering Workshops, pages 206-211, April 11-16 2011	published

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Windsor. I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses

the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis. I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Web Recommendation Systems (*WRS's*) are used to recommend items and future page views to world wide web users. Web usage mining lays the platform for WRS's, as results of mining user browsing patterns are used for recommendation and prediction. Existing WRS's are still limited by several problems, some of which are the problem of recommending items to a new user whose browsing history is not available (Cold Start), sparse data structures (Sparsity), and no diversity in the set of recommended items (Content Overspecialization). Existing WRS's also fail to make full use of the semantic information about items and the relations (e.g., *is-a*, *has-a*, *part-of*) among them. A domain ontology, advocated by the Semantic Web, provides a formal representation of domain knowledge with relations, concepts and axioms.

This thesis proposes *SemAware* system, which integrates domain ontology into web usage mining and web recommendation, and increases the effectiveness and efficiency of the system by solving problems of cold start, sparsity, content overspecialization and complexity-accuracy tradeoffs. *SemAware* technique includes enriching the web log with semantic information through a proposed semantic distance measure based on Jaccard coefficient. A matrix of semantic distances is then used in Semantics-aware Sequential Pattern Mining (*SPM*) of the web log, and is also integrated with the transition probability matrix of Markov models built from the web log. In the recommendation phase, the proposed SPM and Markov models are used to add interpretability. The proposed recommendation engine uses vector-space model to build an item-concept correlation matrix in combination with user-provided tags to generate top-n recommendation.

Experimental studies show that *SemAware* outperforms popular recommendation algorithms, and that its proposed components are effective and efficient for solving the contradicting predictions problem, the scalability and sparsity of SPM and top-n recommendations, and content overspecialization problems.

"In the name of Allah, the Beneficent, the Merciful (1)

Praise be to Allah, Lord of the Worlds, (2)"

Quran 1-2:1

Acknowledgements

I stand speechless and humbled in front of my wife who sacrificed everything, threw herself into the unknown, and always stood next to me, without you I will never be who I am now.

I would like to acknowledge the important role of my PhD committee and the External Reviewer, and thank them for their enlighting and encouraging comments and reviews. Most of all my advisor Prof. Christie Ezeife who guided me thorough different hardships during my study and provided unsurpassed mentoring and support. To all the faculty and staff at the School of Computer Science I bow in respect.

To my parents, I owe you everything and I can never fully pay you back whatever I say or do. I love you and I miss you even when you are next to me. My kids, this is for you; Nada the always proud of her parents, Karine the explorer who always looked up to me, and little Basel who always asked to see videos of us playing at the beach while I am writing this thesis!

I want also to extend my regards and thanks to my friends and colleagues Tariq El-Amsy, Abdul Al-Khateb, Jalal Al-Sholi, Hijaz Al-Ani, Zina Ibrahim and Dan Zhang. Fuad Enaya, Raed Annabosi and friends in KFUPM, I would not have made it without your prayers and friendship. Finally, to my in-laws who always believed in me and provided all the support they can, and my students at UWin and KFUPM, thank you.

Contents

List of Tables

List of Figures

List of Abbreviations

_Min_sup_ Minimum support, a user specified threshold of how frequent a sequence should be (a minimum limit), also denoted as $\sigma(S)$ or ζ

CB Content-Based web recommendation systems

CF Collaborative Filtering web recommendation systems

CID Customer ID

CRM Customer Relationship Management

DL Description Logic

FP-SMM Frequency-Pruned Selective Markov Model

FSS Feature Subset Selection

GPS Geographic Positioning System

HTML Hyper Text Markup Language

HTTP Hypter Text Transmission Protocol

IMDb Internet Movies Database

IP Internet Protocol

ISP Internet Service Provider

LCA Lowest Common Ancestor

ODP The Open Directory Project http://www.dmoz.org/

OWL Web Ontology Language

PLSA Probabilistic Latent Semantic Analysis

RDF Resource Description Framework

SID Session ID

SMM Selective Markov Model

SP-SMM Semantics-Pruned Selective Markov Model

SPM Sequential Pattern Mining

SR-FPSMM Semantics-Rich Frequency-Pruned Selective Markov Model

SR-SMM Semantics-Rich Selective Markov Model

SR-SP-SMM Semantics-Rich Semantics-Pruned Selective Markov Model

TID Transaction ID

URI Uniform Resource Identifier

URL Uniform Resource Locator

W3C World Wide Web Consortium

WR Web Recommendation

WRS Web Recommendation System, also used to refer to the plural form in some parts of the text

WUM Web Usage Mining

Chapter 1

Introduction

RECOMMENDATION SYSTEMS are being used more than ever in a wide range of services, such as financial investments, medical and e-commerce applications. In e-commerce, Web Recommendation Systems (WRS) rely on the history and behavior of users to recommend future item purchases and page views, these systems are built on top of Web Usage Mining.

Web usage mining is concerned with finding user navigational patterns on the world wide web by extracting knowledge from web usage logs (we will refer to them as web logs). The assumption is that a web user can physically access only one web page at any given point in time, that represents one item. Finding frequent user's web access sequences is done by applying Sequential Pattern Mining (SPM) [Agrawal and Srikant, 1995] techniques on the web log. SPM fits the problem of mining the web log directly. On the other hand, current sequential pattern mining techniques suffer from a number of drawbacks [Mabroukeh and Ezeife, 2010], most importantly they do not incorporate semantic information into the mining process and do not provide a way for predicting future user access patterns or, at least, user's next page request, as a direct result of mining. Predicting user's next page request (referred to as *Prediction* or *Page Prefetching*), usually takes place as an additional phase after mining the web log.

The process of Web Usage Mining goes through the following three phases. (1) Pre-processing phase: The main task here is to clean up the web log by removing noisy and irrelevant data. In this phase also, users are identified and their accessed web pages are organized sequentially into sessions according to their access time, and stored in a sequence database. (2) Pattern Discovery phase: The core of the mining process is in this phase. Usually, Sequential Pattern Mining (SPM) is used against the cleaned web log to mine all the frequent sequential patterns. (3) Recommendation/Prediction phase: Mined patterns

and probabilistic models are used to generate recommendations.

Web Recommendation Systems (also called Web Recommenders) rely on results of mining the web log, of a single or several users, to recommend future page views or products to the active user (who is currently browsing the web site). One kind of WRS's is *Collaborative Filtering* in which classification is used as a way for mining the web log and categorizing user profiles, then recommending a product or web page for a user as rated or mostly used by the category of users to which his profile belongs. Researchers focusing on WR and web personalization try to improve their systems by incorporating text and semantic information from web pages, and using it together with usage data [Dai and Mobasher, 2003]. On the other hand, domain knowledge is not being used in all of the phases of Web Usage Mining and WR, and not all the axioms and power that a domain ontology can offer are being utilized. Yet, WRS's still suffer from problems of cold start, content overspecialization, and scalability. The integration of the ontological information representing the underlying concepts, and attributes embedded in a site allows for more effective pattern discovery and solves important problems in WRS's. In addition, this integration allows systems to infer on concept generalizations, and provide interpretability to explain and reason about user behavior.

1.1 Motivation

As the Semantic Web becomes more endowed with specific standards, more businesses on the Internet are starting to include domain ontologies in their online applications (e.g., Amazon.com[1], eBay[2]), due to the continuous development and use of Semantic Web and Web 2.0 technologies[3]. It is provided as an underlying ontology for several web applications (like the Internet Movie Database $IMDb$[4]). Tagging and semantic annotation of web pages is also spreading widely on the world wide web, towards realizing the semantic web (e.g., Flickr[5], delicious[6], YouTube[7]). An ontology provides a set of well-founded constructs that define significant concepts and their semantic relationships. Such constructs can be leveraged to build meaningful higher level knowledge in a particular domain. Consider an

---

[1]http://www.wsmo.org/TR/d3/d3.4/v0.2/#ontology

[2]http://www.ebay.com

[3]we refer to Tim O'Reilly's definition of Web 2.0 (at http://oreilly.com/web2/archive/what-is-web-20.html), that includes social networks and tagging systems made possible by new web development and user interaction tools.

[4]http://www.imdb.com

[5]http://www.flickr.com

[6]http://www.delicious.com

[7]http://www.youtube.com

e-commerce application on the Internet that sells electronic items, call it *eMart*. It has a domain ontology of all concepts of electronic items built by the ontology engineer. This application allows its users to put up items for sale on its web site. eMart wants to be able to match the active user's interests with an appropriate electronic item for purchase, also it wants to be able to predict what the next item a user will view based on his browsing history. Users in eMart are allowed to tag items they browse or sell. As the active user (call him John) arrives at eMart's web site and is browsing a page that describes a GPS device, next he moves to a page that describes a digital camera, then is looking at a camera lens which he tags with some keywords, then he moves to browse and add to his cart a special lens filter. As John is tagging some items, the system is taking note that those items are of interests to him. Another first-time user (call her Jane) arrives at eMart and initiates a search using keywords to describe a product of interest to her. For John, eMart is able to provide a set of recommended items to browse, based on his browsing history. As for Jane, eMart is able to match the keywords from her search query (alternatively she could have saved a set of keywords in her user profile at eMart) with tags that users similar to John have provided, to generate a set of top-n recommended items that match her query.

Without a domain ontology eMart is not able to categorize products or find relationships between them. Also it is not able to calculate similarities between products or similarity between Jane's query and products stored in eMart's catalog. It is not possible for eMart to answer questions like "Why did you recommend an SD card to me?" without using axioms provided by a domain ontology, or answer queries like "What *new* items can I be interested in?", or "What other kinds of items can interest me?"

1.2   Contributions of Thesis

This thesis shows that domain knowledge (in the form of domain ontology with relations) can be integrated into all phases of Web Usage Mining and Web Recommendation Systems to improve their effectiveness, to provide meaningful recommendations and to solve problems in these systems. Table 1.1 shows a list of all algorithms proposed in this thesis with brief description of each. This thesis and published work originating from it [Mabroukeh and Ezeife, 2009a,b, 2010, 2011] contribute to research as follows in the next two subsections.

1.2.1   Functional Contributions

The functionalities that this thesis adds to existing systems include:

1. The proposed *SemAware* system integrates domain knowledge into all three phases of Web Usage Mining and Recommendation. (algorithms getObject, SemAwareSPM, SP-SMM and SemAwareIN in Table 1.1). Such integration adds interpretability to results of WRS's, at different levels of abstraction and detail of the domain knowledge.

2. Domain ontology is used in Web Recommendation with user-provided tags to enable WRS's to provide top-n recommendations without item clustering, depending on scientific methods from natural language processing and vector-space models in the proposed algorithm SemAwareIN.

3. Efficiency of SPM algorithms is enhanced by the proposed Semantics-aware SPM in algorithm SemAwareSPM. Execution time is reduced by more than 80%, and memory consumption is reduced by about 57%.

4. A novel method is introduced for enriching the Markov model's transition probability matrix with semantic information for next item prediction (algorithms SP-SMM and Semantic-rich 1st-order Markov in Table 1.1), by a proposed combination function $\oplus$. This combination solves the problem of contradicting predictions, and contributes towards minimizing complexity-accuracy tradeoff.

1.2.2   Procedural Contributions

Different procedural contributions are proposed in this thesis to achieve the discussed functionalities. These contributions are:

1. In order to solve the problems discussed and to increase effectiveness of WRS's, the thesis stresses on the use of full domain ontology rather than simple hierarchical taxonomy, by proposing a measure which incorporates all semantic relations of the ontology in order to compute semantic distance between any two ontology concepts, based on Jaccard coefficient [Jaccard, 1901]. This measure is used to achieve functionalities (1), (3) and (4).

2. To achieve functionalities (1) and (2), and to solve limitations of WRS's (namely, the cold start, sparsity and scalability, and content overspecialization problems), we use scientific methods of SPM, cosine similarity, conditional probability and spreading activation in algorithms SemApJoin, Assoc2Markov, SemAwareIN and SemAwareIN-Ex (Table 1.1). Chapter 3 provides a detailed description of these problems and the proposed solutions.

Table 1.1: List of algorithms proposed by this thesis.

SemAware Phase	Algorithm Name	Brief Description
Preprocessing	OntoCraft	To build a simple domain ontology from the web log
	getObject	Maps a semantic object from a web page to its ontology concept
Pattern Discovery / Mining Core	SemAwareSPM	General algorithm for semantics-aware Apriori-based SPM
	SemApJoin	A semantics-aware join procedure for AprioriAll-sem
	PLWAP-sem	A semantics-aware variation of PLWAP [Ezeife and Lu, 2005]
	Semantic-rich 1st-order Markov	A method to integrate semantic distance with Markov transition probabilities
	SP-SMM	Semantically-pruned Selective Markov models
Postprocessing	Ontology_RollUp	For ontology-based association rules generalization
	Ontology_DrillDown	For ontology-based association rules specialization
	Assoc2Markov	Complements association rules with results from SMM
	SemAwareIN	Ontology-based top-n WR from user tags
	SemAwareIN-LCA	ontology-based dimensionality reduction method using Lowest Common Ancestor
	SemAwareIN-FSS-50	WR with dimensionality reduction using Feature Subset Selection
	SemAwareIN-Ex	Recommendation Expansion algorithm that relies on ontological relation

3. Algorithm Assoc2Markov is proposed to combine results of Markov models and association rules for more accurate recommendation. This contributes towards achieving functionality (1) and solving the problem of contradicting predictions, using the scientific method of conditional probability and confidence measures.

4. A novel method is proposed for dimensionality reduction in top-n recommendation

using the domain ontology and the concept of Lowest Common Ancestor, this method is compared with subset feature selection technique used to increase the scalability of this system (algorithms SemAwareIN-LCA and SemAwareIN-FSS-50 in Table 1.1).

5. In the absence of domain ontology, algorithm OntoCraft is proposed to build a basic domain ontology from the web log of user accesse sequences. SemAware uses this algorithm in its preprocessing phase to achieve functionality (1).

6. A comprehensive survey of Sequential Pattern Mining (SPM) algorithms is carried out with experimental comparison of top algorithms. The proposed taxonomy for algorithms is the first one in the area that presents a hierarchical, a tabular and a chronological ordering of the algorithms along with their features [Mabroukeh and Ezeife, 2010]. It provides a deep understanding of the different algorithms, their component features, and the different techniques and methods used in research so far, highlighting comparative advantages and drawbacks of these algorithms.

1.3 Thesis Outline

Definitions and survey of related work are presented in Chapter 2. This thesis poses a set of research questions in the area of sequential pattern mining and content-based WRS's. The thesis hypothesis is formulated based on these questions and discussed in Chapter 3 along with proposed solutions and the proposed SemAware system. Enriching the web log with semantic information is discussed in Chapter 4, which also presents data preparation for semantics-aware mining and recommendation. The preprocessing algorithms *OntoCraft* and *getObject* are presented in the same chapter. Chapter 5 proposes to use semantic information in the pattern discovery phase, where we present the semantics-aware algorithms *AprioriAll-sem*, *GSP-sem* and *PLWAP-sem*. Then, Chapter 6 shows how such semantic infromation is used in a probabilistic model by building and pruning semantic-rich and Selective Markov Models for next page request prediction. Web recommendation (the post-processing in SemAware) is discussed in Chapter 7, that proposes to use domain ontology for recommendation by *SemAwareIN* algorithm to solve the problems and limitations of content-based web recommendation, and proposes also *SemAwareIN-LCA*, for dimensionality reduction using the domain ontology. Other proposed variations of *SemAwareIN* are also discussed in Chapter 7 and compared with state-of-the-art WRS algorithms

Experimental evaluations and results accompany each of the mentioned chapters as a dedicated section. Finally, Chapter 8 concludes this thesis with important remarks and future research directions.

Chapter 2

Related Work

This chapter presents a detailed introduction to the topic with required definitions. It starts first by explaining the concept of Web Usage Mining and the web log, presents problem definition for sequential pattern mining and surveys the area. Afterwards, definition of the domain ontology used in the proposed system is provided with examples. The chapter then looks at Web Recommendation Systems (WRS) and proceeds to provide a definition of ontology-based WRS's, after which the topic is surveyed identifying pitfalls and problems.

2.1   Web Usage Mining

Web usage mining (also called *web log mining*) is an important application concerned with finding user navigational patterns on the world wide web by extracting knowledge from web logs, where ordered sequences of events in the sequence database are composed of single items and not sets of items, with the assumption that a web user can physically access only one web page at any given point in time. If a time window of access is considered, which may allow a web user to browse a collection of web pages over a specified period of time, then, it reverts back to a general sequence database. Currently, most web usage mining solutions consider web access by a user as one page at a time, giving rise to special sequence database with only one item in each sequence's ordered event list. Thus, given a set of events $E = \{a, b, c, d, e, f\}$, which may represent product web pages accessed by users in eMart (an example e-commerce application), a web access sequence database for four users may have the four records: $[t_1, < abdac >]$; $[t_2, < eaebcac >]$; $[t_3, < babfaec >]$; $[t_4, < abfac >]$. A web log pattern mining on this web sequence database can find a frequent sequence $< abac >$ indicating that over 90% of users who visit product a's web page of http://www.eMart.com/producta.htm also immediately visit product b's web page

of http://www.eMart.com/productb.htm and then revisit product a's page, before visiting product c's page. Store managers may then place promotional prices on product a's web page, that is visited a number of times in sequence, to increase the sale of other products. To mine such sequences, web usage mining relies on Sequential Pattern Mining (SPM) of the web log [Pei et al., 2000; Ezeife and Lu, 2005; El-Sayed et al., 2004; Wang and Han, 2004; Goethals, 2005]. Typical applications of web usage mining fall into the area of user modeling, such as web content personalization, web site reorganization, pre-fetching and caching, recommendation, e-commerce and business intelligence [Facca and Lanzi, 2005].

2.2 The Web Log and Data Preprocessing

The web log is a registry of web pages accessed by different users at different times, which can be maintained at the *server-side*, *client-side* or at a *proxy server*, each having its own benefits and drawbacks on finding the users' relevant patterns and navigational sessions [Iváncsy and Vajk, 2006]. A web log stored at the client-side captures only web accesses by that particular client/user and could be beneficial in mining access sequences for a particular user as part of a web personalization system [Fenstermacher and Ginsburg, 2002; Lu et al., 2003]. This requires that a remote agent be implemented or a modified browser be used to collect single-user data, thus eliminating caching and session identification problems. Whereas a proxy-side web log captures access sequences of the clients of a certain service provider company and could be used in applications like page pre-fetching [Pitkow and Pirolli, 1999] and caching to enhance the performance of the proxy server. A proxy server can also reveal the actual HTTP requests from multiple clients to multiple web servers, thus, characterizing the browsing behavior of a group of anonymous users sharing a common server [Srivastava et al., 2000], that can serve the current trend of Group Recommendation Systems. Web access sequences stored on the server-side represent web page accesses of all users who visit this server at all times, which is good for mining multiple users' behavior and for web recommender systems. While server logs may not be entirely reliable due to caching as cached page views are not recorded in a server log. But this problem is solved by referring to logged information to infer and re-construct user paths, filling out missing pages.

Work in this thesis focuses on server-side web logs used by e-commerce web sites. Such web logs are usually raw registries of URLs visited by different users. Figure 2.1 shows some lines extracted from a standard web log[1]. One can notice that each line contains the following data: 1) timestamp of the HTTP request, 2) IP address of the requesting client,

---

[1]This "extended" web log format is a W3C standard http://www.w3.org/TR/WD-logfile.html

3) HTTP request command with requested URI, and 4) HTTP protocol version and client browser details. Clearly, there is no representation whatsoever of domain knowledge or any means of describing the requested products.

```
1 | 2006-02-01 00:08:43 1.2.3.4 - GET /classes/cs589/papers.html - 200 9221
  | HTTP/1.1 maya.cs.depaul.edu
  | Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+.NET+CLR+2.0.50727)
  | http://dataminingresources.blogspot.com/
2 | 2006-02-01 00:08:46 1.2.3.4 - GET /classes/cs589/papers/cms-tai.pdf - 200 4096
  | HTTP/1.1 maya.cs.depaul.edu
  | Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+.NET+CLR+2.0.50727)
  | http://maya.cs.depaul.edu/~classes/cs589/papers.html
3 | 2006-02-01 08:01:28 2.3.4.5 - GET /classes/ds575/papers/hyperlink.pdf - 200
  | 318814 HTTP/1.1 maya.cs.depaul.edu
  | Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1)
  | http://www.google.com/search?hl=en&lr=&q=hyperlink+analysis+for+the+web+survey
4 | 2006-02-02 19:34:45 3.4.5.6 - GET /classes/cs480/announce.html - 200 3794
  | HTTP/1.1 maya.cs.depaul.edu
  | Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1)
  | http://maya.cs.depaul.edu/~classes/cs480/
5 | 2006-02-02 19:34:45 3.4.5.6 - GET /classes/cs480/styles2.css - 200 1636
  | HTTP/1.1 maya.cs.depaul.edu
  | Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1)
  | http://maya.cs.depaul.edu/~classes/cs480/announce.html
6 | 2006-02-02 19:34:45 3.4.5.6 - GET /classes/cs480/header.gif - 200 6027
  | HTTP/1.1 maya.cs.depaul.edu
  | Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1)
  | http://maya.cs.depaul.edu/~classes/cs480/announce.html
```

Figure 2.1: A sample of six lines from a typical web log, extracted from Mobasher [2006].

Before entering the mining phase, the web log goes through a four-step preprocessing phase [Meo et al., 2004] that includes *cleaning* and *sessionizing*. During cleaning, any HTTP request not relevant to the mining process is deleted, like requests for images or other media embedded in the HTML page or webspiders' navigation, thus, decreasing the size of the log. Sessionizing refers to the identification of different users' sessions, each representing a single visit to the site. Preprocessing also tries to distinguish among different users (what can be referred to as *user identification*). A common approach to distinguish among unique visitors is the use of packet sniffer and client-side cookies. Not all sites, however, employ cookies, and due to privacy concerns, client-side cookies are sometimes disabled by users. IP addresses, alone, are not generally sufficient for mapping log entries onto the set of unique visitors and their sessions. This is mainly due to the proliferation

of ISP proxy servers which assign rotating IP addresses to clients as they browse the web. Using IP addresses has been the area of debate and much research [Pabarskaite and Raudys, 2007; Huntington et al., 2008]. The end product of preprocessing phase is the navigational patterns of each user extracted as a chronological sequence of numbers or alphabetic characters, where each of these numbers/characters denotes a web page accessed by that user. A user can have more than one sequence corresponding to several web surfing sessions.

In most cases, researchers are assuming that user web visit information is completely recorded in the web server log, which is preprocessed to obtain the transaction database to be mined for sequences [Ezeife and Lu, 2005]. According to W3C [1999], a *user session* is defined as "a delimiting set of user clicks across one or more web servers", and a *server session* is defined as "a collection of user clicks to a single web server during a user session". While a *pageview* is "the visual rendering of a web page in a specific environment at a specific point in time", think of it as a single web page p_i from the set $\mathbb{P}$ of all unique web pages in the application under consideration, $p_i \in \mathbb{P}$. A *clickstream* is defined as a sequential series of pageview requests made by a single user, referred to as a transaction t_j, such that $t_j = \{p_1 p_2 \ldots p_n\}$, where $n \leq |\mathbb{P}|$.

Definition 1 (Web Log). *A web log $\mathcal{W}$ is a set of clickstream transactions, such that $\mathcal{W} = \{t_1, t_2, \ldots, t_j, \ldots, t_l\}$.* □

A web page p_i in a pageview represents a certain product or item of interest to the user. Thus we use p_i to refer to a pageview or an item interchangeably throughout this thesis.

2.3 Sequential Pattern Mining

This section is part of a full survey published in the journal of ACM Computing Surveys [Mabroukeh and Ezeife, 2010].

Sequential pattern mining discovers frequent subsequences as patterns in a sequence database. After preprocessing the web log, all sequences for all customers are stored in a Sequence Database along with Customer ID *CID* and Transaction/Session ID *TID*. A sequence database stores a number of records, where all records are sequences of ordered events, with or without concrete notions of time. An example sequence database is retail customers' transactions or purchase sequences in a grocery store showing for each customer, the collection of store items they purchased every week for one month. These sequences of customer purchases can be represented as records with schema [Transaction/Customer ID,

<Ordered Sequence Events>], where each sequence event is a set of store items like bread, sugar, tea, milk. For only two such customers, an example purchase sequential database is $[t_1, <$(bread, milk), (bread, milk, sugar), (milk), (tea, sugar)$>]$; $[t_2, <$(bread), (sugar, tea)$>]$. While the first customer, with transaction ID shown as t_1 in the example, made a purchase each of the four weeks, the second customer represented by t_2 made purchases only two of the weeks. Also, a customer can purchase one or more items during each market visit. Thus, records in a sequence database can have different lengths and each event in a sequence can have one or more items in its set. Other examples of sequences are for DNA sequences, stock market trends and web log data. Sequential pattern mining is an important data mining problem with broad applications, including the analysis of customer purchase behavior, web access patterns, scientific experiments, disease treatments, natural disaster, and sequences of amino acids in proteins. A sequential pattern mining algorithm mines the sequence database looking for repeating patterns (known as *frequent sequences*) that can be used later by end users or management to find associations between the different items or events in their data for purposes such as marketing campaigns, business intelligence, prediction and planning.

Definition 2 (Sequential Pattern Mining). *Given (i) a set of sequential records (called sequences) representing a sequential database D, (ii) a minimum support threshold called min_sup ξ, (iii) a set of k unique items or events $I = \{i_1, i_2, \ldots, i_k\}$, the problem of mining sequential patterns is that of finding the set of all frequent sequences S in the given sequence database D of items I at the given min_sup ξ.* $\square$

For example, in a web usage mining domain, the items in I can represent the set $\mathbb{P}$ of all web pages (e.g., pages a, b, c, d, e, f) or products (e.g., TV, radio) being sold in an e-commerce web site. An itemset is a non-empty, unordered collection of items (e.g., *(abe)* which are accessed at the same time). A sequence is a lexicographically ordered list of itemsets, e.g., $S = < a(be)c(ad) >$. *Set Lexicographic Order* [Rymon, 1992] is a total linear order which can be defined as follows. Assume an itemset t of distinct items, $t = \{i_1, i_2, \ldots, i_k\}$, and another itemset of distinct items also $t' = \{j_1, j_2, \ldots, j_l\}$, where $i_1 \leq i_2 \leq \cdots \leq i_k$ and $j_1 \leq j_2 \leq \cdots \leq j_l$, such that $\leq$ indicates "occurs before" relationship. Then, for itemsets, $t < t'$ (t is lexicographically less than t') iff either of the following is true:

1. for some integer h, $0 \leq h \leq min\{k,l\}$, we have $i_r = j_r$ for $r < h$, and $i_h < j_h$, or

2. $k < l$, and $i_1 = j_1$, $i_2 = j_2$,...,$i_k = j_k$.

An example of the first case is $(abc) < (abec)$ and $(af) < (bf)$, the second case is similar to a strict subset relationship, where t is a strict subset of t', for example, $(ab) < (abc)$.

An itemset is a set drawn from items in I, and denoted as $(i_1, i_2, \ldots, i_k)$, where i_j is an item or event. A sequence S is denoted as a sequence of items or events $< e_1 e_2 e_3 \ldots e_q >$, where the sequence element e_j is an itemset (e.g., (be) in $<a(be)c(ad)>$) that might contain only one item (which is also referred to as 1-itemset). A sequence with k items is called a k-sequence. An item can occur only once in an itemset but it can occur several times in different itemsets of a sequence. A sequence $\alpha = < e_{i_1} e_{i_2} e_{i_3} \ldots e_{i_m} >$ is a subsequence of another sequence $\beta = < e_1 e_2 e_3 \ldots e_n >$ denoted as $\alpha \preceq \beta$, if there exists integers $i_1 < i_2 < \cdots < i_m$ and all events $e_{i_j} \in \alpha$, and $e_i \in \beta$, and $i_1 \leq 1$ and $i_m \leq n$, such that $e_{i_j} \subseteq e_i$. A sequential pattern is *maximal* if it is not a subsequence of any other sequential pattern. Sequence lexicographical ordering can be defined as follows. Assume a lexicographical order $\leq$ of items I in the sequential access database, denoted as $\leq_I$. If an item i occurs before an item j, it is denoted as $i \leq_I j$, this order is also extended to sequences and subsequences by defining $S_a \leq S_b$ if S_a is a subsequence of S_b. Consider all sequences arranged in a sequence tree (referred to as Lexicographical Tree) T, as in Figure 2.2 as follows: The root of the tree is labeled $\{\}$. Recursively, if n is a node in the tree T, then n's children are all nodes n' such that $n \leq n'$ and $\forall m \in T : n' \leq m \implies n \leq m$ each sequence in the tree can be extended by adding a 1-sequence to its end or adding an itemset to its end, in the former case it is called a *sequence-extended sequence* and in the later case it is an *itemset-extended sequence*, which is not applicable to the case of web log mining.
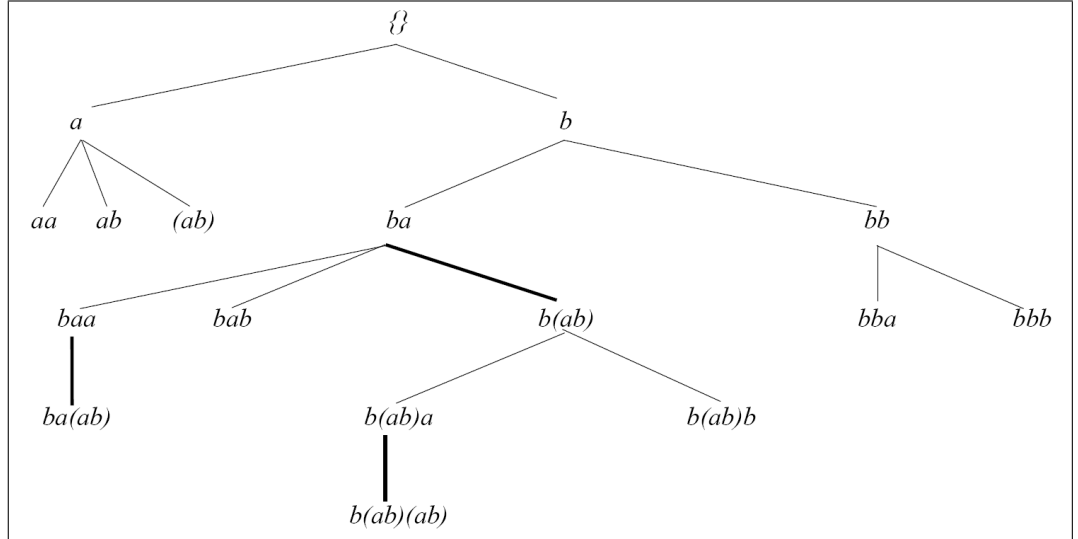
Figure 2.2: Lexicographic sequence subtree for only a and b items. Light lines represent sequence-extended sequence, bold lines represent item-set extended sequence.

The frequency or support of a sequence (or subsequence) S, denoted as $\sigma(S)$, is the total

number of sequences of which S is a subsequence divided by the total number of sequences in the database D, whereas, the *absolute support* (or *support count*) of a sequence (or subsequence) S, is the total number of sequences in D of which S is a subsequence. A sequence is called *frequent* if its frequency is not less than a user-specified threshold called minimum support, denoted as, min_sup or the greek letter ξ. A frequent sequence S_α is called a frequent *closed sequence* if there exists no proper supersequence of S_α with the same support, i.e., $\nexists S_\beta$ such that $S_\alpha \preceq S_\beta$ and $\sigma(S_\alpha) = \sigma(S_\beta)$, otherwise, it is said that sequence S_α is absorbed by S_β [Wang and Han, 2004]. For example, assume the frequent sequence $S_\beta = <beadc>$ is the only superset of the frequent sequence $S_\alpha = <bea>$, if $\sigma(S_\alpha) = \sigma(S_\beta)$ then S_α is not a frequent closed sequence, on the other hand, if $\sigma(S_\alpha) > \sigma(S_\beta)$, then S_α is a frequent closed sequence. Notice that $\sigma(S_\beta)$ cannot be greater than $\sigma(S_\alpha)$, because $S_\alpha \preceq S_\beta$. In web usage mining, D is a sequence database of transactions representing web accesses, where each transaction t has a unique identifier (Transaction id (TID) or Session id (SID)) and a sequence of single-element set events as in $<bcabd>$ and $<bcabdac>$. The problem of sequential pattern mining is restricted to sequential web log mining with a D, min_sup ξ, and set of events $E = \{a, b, c, d, \ldots\}$ representing pageviews, with the following characteristics:

1. Patterns in a web log consist of contiguous page views (items in the sequences). No two pages can be accessed by the same user at the same time, and so sequences contain only 1-itemsets (i.e., single items). An example sequence is $<bcabdac>$ which is different from a general sequence like $<(ba)(ab)d(ac)>$.

2. Order is also important in web usage mining as the order of page references in a transaction sequence is important. Also, each event or item can be repeated, and it represents page refreshes and backward traversals (e.g., $<aba>$, $<aab>$, where event a is repeated in these sequences).

3. Web logs are sparse datasets, i.e., usually there are many unique items with only few repetitions of them in the sequences of one user, which makes algorithms targeting sparse datasets (e.g., LAPIN_WEB [Yang et al., 2006]) perform better than those for dense datasets (e.g., PrefixSpan [Pei et al., 2001]), as claimed in Yang et al. [2006]. Our experimental results, however, show that LAPIN_Suffix [Yang et al., 2005], outperforms PrefixSpan only on large datasets and at higher support when there are likely not many frequent patterns.

Several sequential pattern mining techniques that can be applied to web usage mining have been introduced in the literature since mid 1990's. Previous surveys looked at different

mining methods applicable to web logs [Srivastava et al., 2000; Facca and Lanzi, 2003; Teisseire and Poncelet, 2005; Facca and Lanzi, 2005; Iváncsy and Vajk, 2006], but they lack of three important things; they fail to (i) focus on sequential patterns as a complete solution, (ii) provide a taxonomy and (iii) include a deep investigation of the techniques and theories used in mining sequential patterns.

2.3.1   A Taxonomy of Sequential Pattern Mining Algorithms

A taxonomy of existing sequential pattern mining algorithms is provided in Mabroukeh and Ezeife [2010] and summarized in Figure 2.3, which lists the algorithms, showing a comparative analysis of different important features in them. Sequential pattern mining algorithms can be categorized into Apriori-based, pattern-growth and early-pruning algorithms. Frequent sequential pattern discovery can essentially be thought of as association rule discovery over a temporal database. While association rule discovery [Agrawal et al., 1993] covers only intra-transaction patterns (itemsets), sequential pattern mining also discovers inter-transaction patterns (sequences), where ordering of items and itemsets is very important, such that the presence of a set of items is followed by another item in a time-ordered set of sessions or transactions. The set of all frequent sequences is a superset of the set of frequent itemsets. From this similarity, the earlier sequential pattern mining algorithms were derived from association rules mining techniques. The first of such sequential pattern mining algorithms is AprioriAll algorithm [Agrawal and Srikant, 1995], derived from the Apriori algorithm [Agrawal and Srikant, 1994]. An algorithm can fall into one or more (hybrid algorithm) of the categories in the proposed taxonomy. Mainly, algorithms differ in two ways:

1. The way candidate sequences are generated and stored. The main goal here, is to minimize the number of candidate sequences generated so as to minimize I/O cost.

2. The way support is counted and how candidate sequences are tested for frequency. The key strategy here, is the ability to eliminate any database or data structure that has to be maintained all the time only for support counting purposes.

   The data structures used to store candidate sequences have also been a research topic and an important heuristic for memory utilization. Usually, a proposed algorithm also has a proposed data structure to accompany it, like SPADE [Zaki, 1998] with vertical databases, PrefixSpan [Pei et al., 2001] with projected databases, WAP-Mine [Pei et al., 2000] with WAP-tree, and PLWAP [Ezeife and Lu, 2005] with its PLWAP-tree.

Algorithm	Apriori-Based		Pattern-Growth								Early-Pruning		
	Generate-and-test	multiple scans of the database	sampling and/or compress-ion	candidate sequence pruning	search space partition-ing	tree project-ion	depth-first traversal	suffix growth	prefix growth	memory-only	support counting avoidance	vertical projection of the DB	position coded
AprioriAll	X	X	X		X								
GSP	X	X		X									
SPADE	X			X	X		X					X	
FreeSpan					X					X			
WAP-mine*			X		X	X		X		X			
PrefixSpan				X	X				X	X			
SPAM	X						X			X		X	
FS-Miner*			X	X	X	X	X	X					
DISC-all				X	X		X		X		X	X	X
Apriori-GST*	X		X	X									
PLWAP*			X			X	X		X				X
HVSM	X			X							X	X	X
LAPIN*				X	X		X		X	X	X	X	X

Figure 2.3: Tabular taxonomy of sequential pattern mining algorithms.
*algorithm is specifically for web log mining or single itemset sequence, or a variation of it for web log mining is provided by the same authors.

2.3.2 Apriori-based Algorithms

Apriori [Agrawal and Srikant, 1994] and *AprioriAll* [Agrawal and Srikant, 1995] set the
basis for a breed of algorithms that depends largely on the Apriori property and uses the
Apriori-generate join procedure to generate candidate sequences. The Apriori property
states that *"All nonempty subsets of a frequent itemset must also be frequent"*. It is also
described as being anti-monotonic (or downward-closed), in that, if a sequence cannot
pass the minimum support test, all of its supersequences will also fail the test. Given a
database of web access sequences $\mathcal{D}$ over $\mathcal{I}$ items, and two sets $X, Y \subseteq \mathcal{I}$, then $X \subseteq Y \implies$
$support(Y) \leq support(X)$, Hence, if a sequence is infrequent, all of its supersets must
be infrequent, and vice versa, if a sequence is frequent, all its subsets must be frequent
too. This anti-monotonicity is used for pruning candidate sequences in the search space,
and is exploited further for the benefit of most pattern-growth algorithms. Apriori-based
algorithms scan the database several times to find frequent itemsets of size k at each kth-
iteration, then perform an exhaustive join procedure to generate a large set of candidate
sequences. Candidate sequences that do not satisfy the Apriori property are pruned, and
so on until there are no more candidate sequences. These techniques suffer from increased
delay in mining, as the number of sequences in the database gets larger. Given n frequent
1-sequences and $min\_sup = 1$, Apriori-based algorithms generate $n^2 + \binom{n}{2}$ candidate 2-
seuqneces and $\binom{n}{3}$ candidate 3-sequences, and so on. Eventually, the total number of
candidate sequences generated will be greater than $\sum_{k=1}^{n} \binom{n}{k}$ (exponential complexity).
Such algorithms that depend *mainly* on the Apriori property, without taking further actions
to narrow the search space have the disadvantage of maintaining the support count for each
subsequence being mined and testing this property during each iteration of the algorithm,
which makes them computationally expensive. To overcome this problem, algorithms have
to find a way to calculate support and prune candidate sequences without counting support
and maintaining the count in each iteration. Most of the solutions provided so far for
reducing the computational cost resulting from the Apriori property, use bitmap vertical
representation of the access sequence database [Zaki, 1998; Ayres et al., 2002; Song et al.,
2005; Yang and Kitsuregawa, 2005] and employ bitwise operations to calculate support at
each iteration. The transformed vertical databases on their turn, introduce overheads that
lower the performance of the proposed algorithm, but not necessarily worse than that of
pattern-growth algorithms. Breadth-first search, generate-and-test and multiple scans of
the database, are all key features of Apriori-based methods, that pose challenging problems
hindering the performance of these algorithms.

2.3.3 Pattern-Growth Algorithms

Soon after the Apriori-based methods of the mid 1990s, and in the early 2000s, pattern growth method emerged as a solution to the problem of generate-and-test. The key idea is to avoid the candidate generation step altogether, and to focus the search on a restricted portion of the initial database. Search space partitioning feature plays an important role in pattern-growth. Almost every pattern-growth algorithm starts by building a representation of the database to be mined, then proposes a way to partition the search space, and generates as less candidate sequences as possible by growing on the already mined frequent sequences, and applying the Apriori property as the search space is being traversed recursively looking for frequent sequences. The early algorithms started by using *projected databases*, e.g. FreeSpan [Han et al., 2000], PrefixSpan [Pei et al., 2001] with the latter being the most influential. A subsequence α' of sequence α is called *a projection* of α w.r.t. (with respect to) prefix β if and only if (1) α' has prefix β and (2) there exists no proper supersequence α'' of α' such that α'' is a subsequence of α and also has prefix β [Pei et al., 2001]. PrefixSpan is based on recursively constructing the patterns by growing on the prefix, and simultaneously, restricting the search to projected databases. This way, the search space is reduced at each step, allowing for better performance in the presence of small support thresholds. PrefixSpan is still considered a benchmark and one of the fastest sequential mining algorithms alongside SPADE [Zaki, 2001]. Another algorithm, WAP-mine [Pei et al., 2000] is the first of pattern-growth algorithms to use a physical tree structure as a representation of the sequence database along with support counts, this tree is mined for frequent sequences instead of scanning the complete sequence database in each step. These tree-projection pattern-growth algorithms (e.g., PLWAP [Ezeife and Lu, 2005]) scan the sequence database at most twice, the first scan finds frequent 1-sequences and the second scan builds the tree with only frequent subsequences along with their support. A "header table" is maintained to point at the first occurrence for each item in a frequent itemset, which is later tracked in a threaded way to mine the tree for frequent sequences, building on the suffix or the prefix. Building only on frequent subsequences solves the huge growth problem that exists in Apriori-based algorithms, making the complexity of bulding the tree in the pattern-growth algorithms WAP-mine and PLWAP $O(nl)$, where n is the number of sequences in the sequence database and l is the length of the longest frequent sequence. The time complexity for mining the tree in PLWAP is $O(fp)$, where f is the number of frequent 1-sequences and p is the total number of resulting frequent patterns, but in WAP-mine this is multiplied by $(p - f)$ times needed for constructing the intermediate WAP-trees [Ezeife and Lu, 2005], resulting in polynomial complexity. It is

shown in Dong and Pei [2007] that PrefixSpan has also polynomial complexity.

2.3.4   Early-Pruning Algorithms

Early-Pruning algorithms are emerging in the literature as a new breed for sequential pattern mining. These algorithms utilize some sort of *position induction* to prune candidate sequences very early in the mining process and avoid support counting as much as possible. The rest of the mining process is simple pattern growth.

The idea of position induction is stated in Yang et al. [2006] as follows:

> If an item's last position is smaller than the current prefix position [during mining], the item cannot appear behind the current prefix in the same customer sequence.

These algorithms usually employ a table to track last positions of each item in the sequence and utilize this information for early candidate sequence pruning; as the last position of an item is the key used to judge whether the item can be appended to a given prefix k-sequence or not, thus avoiding support counting and generation of candidate non-frequent sequences. LAPIN [Yang et al., 2007] is comparable to PrefixSpan and is one of the promising algorithms in this category. There are different previous attempts for different mining scenarios, starting with LAPIN-SPAM [Yang et al., 2005] introduced as a solution to overhead problem of bitwise operations in SPAM [Ayres et al., 2002] and claimed to slash memory utilization by half, then LAPIN_LCI, LAPIN_Suffix [Yang et al., 2005] for suffix growth methods and LAPIN_WEB [Yang et al., 2006] for web log mining. In LAPIN-SPAM, Yang and Kitsuregawa [2005] did not conduct experiments or discuss cases of overhead processing delay for preparing the positions table and its compressed *key position* version. Further investigation needs to be done on whether constructing the optimized table introduces start-up delay. While LAPIN is different from LAPIN-SPAM in that it does not use a bitmap representation of the database, Song et al. [2005] in HVSM go further on stressing the importance of bitmap representation, and add to it what they call *first-horizontal last-vertical* database scanning method, then use a special tree structure where each node takes its frequent sibling nodes as its child nodes in the join process to extend the itemset, thus avoiding support counting, but HVSM's performance does not exceed that of SPAM. DISC-all [Chiu et al., 2004] on the other hand, employs temporal ordering besides the regular lexicographic ordering of items and itemsets in the sequence, and uses a variation of projected databases for partitioning the search space depending on the location order of frequent 1-sequences.

2.3.5   Comparative Analysis of Sequential Pattern Mining Algorithms

Table 2.1 shows a comparative performance analysis of algorithms from each of the discussed taxonomy categories. Experimentation was performed on 1.87GHz Intel Core Duo computer with 2 gigabytes memory running Windows Vista 32-bit, with the same implementations of the programs on synthetic data sets generated using the IBM resource data generator code [Agrawal et al., 1993], tested also on a real data set from the School of Computer Science web log. The following parameters are used to generate the data sets as described in Agrawal and Srikant [1995]: |D| represents the number of sequences in the database, |C| is the average length of the sequences, |S| is the average length of maximal potentially frequent sequence, |N| is the number of events, |T| is the average number of items per transaction. Two data sets are used, a medium sized data set described as C5T3S5N50D200K and a large sized data set described as C15T8S8N120D800K. These are run at different minimum support values, low minimum supports of between 0.1% and 0.9% and regular minimum support of 1% to 10%. CPU execution time is reported by the program of each algorithm in seconds, while physical memory usage was measured using Microsoft CLR Profiler. GSP, PLWAP[1], and WAP-mine were initially implemented with C++ language running under Inprise C++ Builder environment and compiled on the command line of MS Visual Studio 9 running on the operating system described above, while the code for SPAM[2], PrefixSpan[3] and LAPIN[4] were downloaded from their respective authors' websites and used as provided. We ran each algorithm alone with dedicated system resources.

Careful investigation of Table 2.1 shows how slow the Apriori-based SPAM algorithm could become as data set size grows from medium (|D|=200K) to large (|D|=800K), due to the increased number of AND operations and the traversal of the large lexicographical tree. Although it is a little faster than PrefixSpan on large data sets, the reason being the utilization of bitmaps as compared to projected databases of PrefixSpan.

One can notice that PLWAP enjoys the fastest execution times, as it clearly separates itself from WAP-mine and PrefixSpan (from the same category of algorithms), especially at low minimum support values when more frequent patterns are found and with large data sets. The version we used for PrefixSpan is the one using *pseudo projection* [Pei et al., 2001] in managing its projected databases. Originally, PrefixSpan scans the whole projected database to find frequent sequences and count their support, the same process is

---

[1]GSP, PLWAP and WAP are available at http://cs.uwindsor.ca/~cezeife/codes.html

[2] http://himalaya-tools.sourceforge.net/Spam/

[3] http://illimine.cs.uiuc.edu/

[4] http://www.tkl.iis.u-tokyo.ac.jp/~yangzl/soft/LAPIN/index.html

Table 2.1: Comparative analysis of algorithms performance. The symbol "-" means an algorithm crashes with provided parameters, and memory usage could not be measured.

	Algorithm	Data set size	Minimum Support	Execution Time (sec)	Memory Usage (MB)
Apriori-based	GSP	Medium ($|D|$=200K)	Low (0.1%)	>3600	800
			Medium (1%)	2126	687
		Large ($|D|$=800K)	Low (0.1%)	-	-
			Medium (1%)	-	-
	SPAM	Medium ($|D|$=200K)	Low (0.1%)	-	-
			Medium (1%)	136	574
		Large ($|D|$=800K)	Low (0.1%)	-	-
			Medium (1%)	674	1052
Pattern-Growth	PrefixSpan	Medium ($|D|$=200K)	Low (0.1%)	31	13
			Medium (1%)	5	10
		Large ($|D|$=800K)	Low (0.1%)	1958	525
			Medium (1%)	798	320
	WAP-mine	Medium ($|D|$=200K)	Low (0.1%)	-	-
			Medium (1%)	27	0.556
		Large ($|D|$=800K)	Low (0.1%)	-	-
			Medium (1%)	50	5
EarlyPruning	LAPIN_Suffix	Medium ($|D|$=200K)	Low (0.1%)	>3600	-
			Medium (1%)	7	8
		Large ($|D|$=800K)	Low (0.1%)	-	-
			Medium (1%)	201	300
Hybrid	PLWAP	Medium ($|D|$=200K)	Low (0.1%)	23	5
			Medium (1%)	10	0.556
		Large ($|D|$=800K)	Low (0.1%)	32	9
			Medium (1%)	21	2

used in WAP-mine with its conditional trees, as each conditional tree presents candidate sequences that require support counting. Manual tracing of a simple sequential pattern mining problem in Mabroukeh and Ezeife [2010], reveals that support counting for frequent sequences in PLWAP entails simply adding up node counts during recursive mining of the virtual conditional subtrees, without having to scan any conditional databases. These findings are reflected on memory consumption during actual runtime of the algorithms on the data sets used for experimentation. It was observed, at that time, that PrefixSpan uses more memory gradually, then it releases it also gradually as mining progresses, while in the

case of WAP-mine more memory gets consumed faster than in PrefixSpan as intermediate trees are created and mined. Then, suddenly memory is all released when recursion has completely unfolded. In the case of PLWAP, more is consumed, also fast, then consumption stays at a stable volume (smaller than WAP-tree) during mining of the tree, then suddenly it is all released at once (not gradually) towards the end of the mining process. This also verifies why projected database techniques use more memory than tree projection (represented by PLWAP and WAP-mine) techniques, as noted in Table 2.1, when small minimum support values are introduced. It is also noticed that PLWAP and WAP-mine keep close execution times to each other for larger minimum support values where there are no frequent patterns mined. This is because the gain in performance by PLWAP mostly occurs when there are more frequent patterns mined. PrefixSpan also shows high speed on data sets with high support when no frequent patterns are found. For example, it only took 0.3sec to mine the medium data set at support of 10%. While LAPIN_Suffix shows performance close to PrefixSpan, it uses less physical memory. PLWAP shows less memory usage than WAP-mine. PrefixSpan, on the other hand uses more memory than any pattern-growth algorithm, even for small data sets as minimum support decreases. It was noticed that it requires at least 2MB of memory just for scanning the sequence database looking for frequent 1-sequences, then memory consumption increases exponentially as projected databases are being created. Most algorithms could not handle mining long sequences with more than 10 events.

In another experiment, the algorithms' scalability was tested as the data set goes from sparse to dense. In this case, four data sets were used, namely, C8T5S4N100D200K (a sparse data set with maximum sequence length of 22 items), C10T6S5N80D200K (a less sparse data set with maximum sequence length of 24 items), C12T8S6N60D200K (a dense data set with maximum sequence length of 31 items), and C15T10S8N20D200K (a more dense data set with maximum sequence length of 33 items). Dense data sets are characterized by having a small number of unique items |N| and a large number of customer sequences |C| and |T|, while sparse data sets are characterized by having a large number of unique items |N| (e.g., a large number of web pages in a given web site) and shorter customer sequences |C| and |T| (e.g., short user browsing sessions). Execution time results (in seconds) are shown in Figure 2.4 at minimum support of 1%. WAP-mine and PLWAP perform faster on sparse data sets than the other algorithms because they employ depth-first search of the projected sparse trees. Also, as the data set gets more dense, PLWAP emerges with least CPU time as no intermediate trees or projected databases are created. GSP also has increased execution time because more frequent 1-sequences occur in dense data sets than in sparse ones.
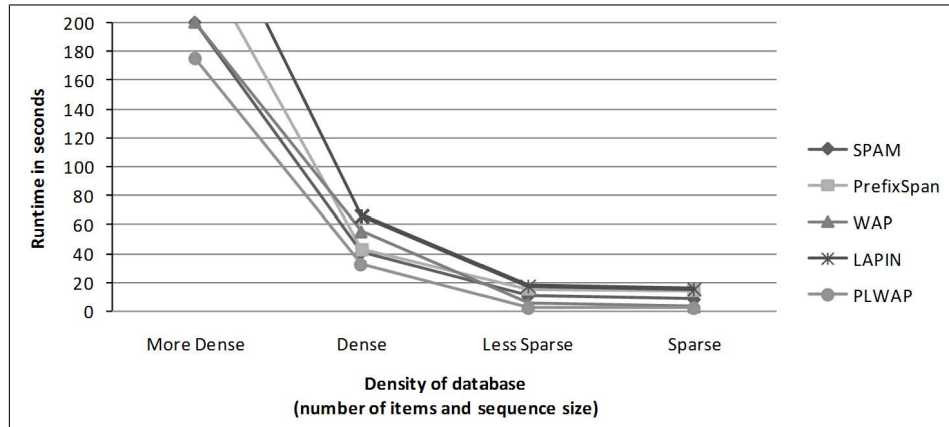
Figure 2.4: Sequence Database density vs. algorithm execution time (in sec), at minimum support of 1%. GSP is not shown as it is out of range of the vertical axis.

2.3.6 Discussion

An investigative look at the preceeding survey [Mabroukeh and Ezeife, 2010] poses the following questions, along with suggested answers:

1. *What are the important features that a reliable sequential pattern mining algorithm should provide?* A reliable algorithm should have acceptable performance measures such as low CPU execution time and low memory utilization when mined with low minimum support values, and should be scalable.

2. *What are the needs of sequential pattern mining applications like web content and web log mining?* The algorithms should be able to handle varying types of data like low quality data with noise, multi-element-set event sequences and single-element-set event sequences, and sparse databases. The techniques should also be extendible to support mining sequence data in other types of domains like distributed, constrained, stream, and object-oriented domains, in addition to multidimensional sequences.

3. *What are the different approaches used so far in general sequential pattern mining and which techniques are suitable for what domains?* The main techniques can be categorized into Apriori-based, pattern-growth, early-pruning and hybrids of these three techniques. Apriori-based algorithms are deemed too slow and have a large search space, while pattern-growth algorithms have been tested extensively on mining the web log and found to be fast, early-pruning have success stories with protein sequences stored in dense databases.

4. *What else can be contributed in sequential pattern mining*? Scalability and handling very long sequences is still a problem not addressed well by many existing techniques. Solutions for distributed and object-oriented domains remain open issues. The ability to mine periodic and time-sensitive frequent patterns, which enables a recommender/prediction system to be context and time-aware, e.g; a group of users might have frequent patterns in an e-commerce web site on holidays and special occasions (like shopping for Christmas) different from their regular frequent patterns. The recommender in a web usage mining system should be able to tell what frequent patterns to mine and utilize based on different contexts. Also, the use of domain knowledge and inclusion of semantic information into the mining process itself requires further attention as presented in this thesis.

This investigation of sequential pattern mining algorithms in the literature reveals that important heuristics employed include, using optimally sized data structure representations of the sequence database, early pruning of candidate sequences, mechanisms to reduce support counting, and maintaining a narrow search space. The quest for finding a reliable sequential pattern mining algorithm should take these points into consideration. It is also noted that literature has recently introduced ways to minimize support counting, although some scholars claim to avoid it [Yang et al., 2005; Wang and Han, 2004; Chiu et al., 2004]. Minimizing support counting is strongly related to minimizing the search space.

In lights of the above, the following points can be considered in a reliable sequential pattern mining algorithm. *First*, a method must generate a search space as small as possible. Features that allow this include early candidate sequence pruning and search space partitioning. Sampling of the database and lossy compression (i.e; concise representation) can also be used to generate a smaller search space. *Secondly*, it is important to narrow the search process within the search space. An algorithm can have a narrow search procedure such as depth-first search. *Thirdly*, methods other than tree projection should be investigated for purposes of finding reliable sequential pattern mining techniques.

2.4   Domain Knowledge and Ontology Representation

The term *domain knowledge* refers to the knowledge associated with a certain concept / topic / discpline which is usually acquired by an expert. Acquiring domain knowledge does not have a standard method or a stand-alone process, knowledge engineers have been using manual techniques that are fit enough for small web sites with single ontologies. For larger websites, machine learning and text mining techniques can be employed to mine the Semantic Web underlying a certain web site. The emergence of XML and RDF (*Resource*

Description Framework) as uniform source identifiers used to describe resources in a web site has provided several notational methods to describe the structure of a document (using XML), and the resources it refers to with their associated properties (using RDF). Large online businesses are starting to use these methods for representation in their web sites to provide a schematic view of the underlying domain knowledge.

An ontology is a formal representation of a domain knowledge, by a set of concepts $\mathcal{C}$ within the domain and the relations $\mathcal{R}$ among them [Stumme et al., 2006].

Definition 3 (Domain Ontology). *A domain ontology with axioms is defined as the structure $\mathcal{O} := (\mathcal{C}, \leq_{\mathcal{C}}, \mathcal{R}, \sigma, \leq_{\mathcal{R}}, \mathcal{A})$ consisting of:*

- *two disjoint sets $\mathcal{C}$ and $\mathcal{R}$ whose elements are called concept identifiers and relation identifiers, respectively, $\mathcal{C}$ is the set of concepts/classes, which are entities in the ontology domain, and $\mathcal{R}$ is a set of relations defined among the concepts,*

- *a partial order $\leq_{\mathcal{C}}$ on $\mathcal{C}$, called concept hierarchy or taxonomy,*

- *a function $\sigma : \mathcal{R} \to \mathcal{C}^{+}$ called signature (where $\mathcal{C}^{+}$ is the set of all finite tuples of elements in $\mathcal{C}$),*

- *a partial order $\leq_{\mathcal{R}}$ on $\mathcal{R}$, called relation hierarchy, and*

- *a set $\mathcal{A}$ of logical axioms in some logical language $\mathcal{L}$, that can describe constraints on the ontology.*

$\square$

Definition 4 (Relation Hierarchy Score). *Define function $\psi_{r_{jk}} : \leq_{\mathcal{R}} \to [0,1]$ that assigns scores to relations $r_{jk} \in \mathcal{R}$ based on their hierarchy $\leq_{\mathcal{R}}$.* $\square$

Figure 2.5 shows part of the ontology for the domain of Cameras and Photography The figure shows ontology classes $\mathcal{C}$ (also called concepts) in oval shapes, like *Camera, Battery*. Each class has a set of attributes. For example, the *Camera* class has attributes *brand, Lens, color*, and the *Video Film* class has attributes *brand, price, length, quality*. Concept hierarchy $\leq_{\mathcal{C}}$ is indicated in the figure with $is-a$ edges. The figure also shows other important relations like *has-a* and *requires*, e.g., one can notice that "Video Camera" *is-a* "Camera", and that a "Camera" *has-a* "Lens". The function σ, maps relations to the set of all finite tuples of elements in $\mathcal{C}$. For example, the relation *has-a* has "Camera" and "Tripod" as signature. The partial order $\leq_{\mathcal{R}}$ on $\mathcal{R}$ represents a flat hierarchy in this

Figure 2.5: Example domain ontology for the domain of Cameras.

example. The set $\mathcal{A}$ of logical axioms is used for inferencing on the logical level. This ontology is manually built using Web Ontology Language OWL[1], and $Prot\acute{e}g\acute{e}$[2], an authoring tool for ontologies. This authoring tool provides a graphical and easy-to-use interface that abstracts the details of *derscription logic* used to build the ontology. Description Logics (DL) are a family of knowledge representation and reasoning formalisms that have been endorsed by the Semantic Web initiative [Berners-Lee et al., 2001]. DLs are based on notions of concepts (unary predicates) and properties (binary relations), such that complex concepts can be built up from primitive ones using different constructs [Hu et al., 2007]. In SemAware we rely on OWL-Lite which is based on $\mathcal{SHIF}$ DL. The preprocessing engine of SemAware parses the ontology as an OWL file using different C# OWL library methods.

Only few WRS's use domain knowledge in their processing, and those that do, mostly, use it in the form of a topic taxonomy (i.e., categorization of items), which we refer to as a *shallow ontology*, where only the *is-a* hierarchical relation is considered. Figure 2.6 shows an example of a topic taxonomy, the taxonomy of Photography from the Open Directory Project (ODP)[3]. The full power of a domain ontology with relations is still to

---

[1]OWL is RDF-based language and a W3C standard, http://www.w3.org/2004/OWL/

[2]http://protege.stanford.edu/

[3]http://www.dmoz.org

```
Accessories ▼
    Bags and Cases
    Batteries and Chargers
    3D
    Lamps and Lighting
    Photofinishing Equipment
Albums and Frames
Backgrounds and Props
Books
Cameras and Camcorders ▼
    Cameras
    Digital ▼
        Accessories ▼
            Batteries
            Printers ▼
                FAQs, Help, and Tutorials
                Manufacturers
                Repairs
                Reviews
                Supplies
        Consumer Information
        Services
        Software
    Repair
Film
Security
Services
Underwater
Used Equipment
```

Figure 2.6: Topic taxonomy of *Photography* from the Open Directory Project ODP, can be found at http://www.dmoz.org/Shopping/Photography.

be utilized in WRS's. This power enables the formulation of more accurate similarity measures between concepts, since concepts share several relations besides the *is-a* relation. Use of ontology with relations also provides better interpretability of recommendation results as this thesis will show. The proposed system in this thesis provides an algorithms to expand the recommendation set (discussed in Section 7.2.2), and to add interpretability to association rules (Section 7.1).

The increased realization of the semantic web and the introduction of page tagging systems, make gathering semantic knowledge from web pages easier. In this case, user-provided tags can be used, saving the trouble of web content mining to generate the keywords. These tags can also be used to annotate the items, if they are not already annotated, using semantic web techniques (for example, in SemAware, OntoMat Annotizer[1]

---

[1]http://annotation.semanticweb.org/ontomat/index.html

is used to annotate the web pages with concepts from the underlying ontology). Once the items are annotated, they can be mapped to the underlying ontology concepts.

2.4.1 Ontology Learning

As the Semantic Web is being realized with all of Web 2.0 technologies of today, the prcoess of learning and building an ontology for an underlying domain from scratch is a teadious, non-standardized and a partially manual process. The framework proposed by this thesis assumes that a domain ontology is provided, but on the other hand algorithm *OntoCraft* (algorithm 1, page 46) is proposed for building a simple ontology from the web log provided as input to SemAware. There are mainly two approaches/techniques for ontology leanring [Maedche et al., 2002]. (a) *Symbolic approaches* based on using lexico-syntactic patterns in a corpus as regular expressions, for the extraction of semantic relations among keywords that are used as ontology concepts. The task here is to define a regular expression that captures patterns that indicate a relation of interest, and map the results to a semantic structure like a taxonomy, an early example from NLP research is CAMILLE [Hastings, 1994] and Hahn and Schnattinger [1998], other examples include the use of heuristic patterns [Hearst, 1998; Caraballo, 2001; Aussenac-gilles, 2005] and extraction from semantically-tagged corpora [Missikoff et al., 2002; Ciravegna et al., 2004]. (b) *Statistics-based approaches* that use probabilities and perform hierarchical clustering of keywords into concepts and categories [Pereira et al., 1993; Glover et al., 2002], that are later refined using classification methods [Manning and Schuetze, 1999]. Some methods use Naïve Bayes [Sanderson and Croft, 1999; Doan et al., 2000]. It is argued in Maedche et al. [2002] that statistics-based approaches allow for better scaling, but symbolic approaches are more precise, although hierarchical clustering requires quadratic time and space complexity.

Zhou [2007] provides a comprehensive survey of state-of-the-art ontology learning methods, in which she refers to symbolic approaches as Rule-Based approaches, and also surveys a hybrid approach that combines between the two discussed approaches (e.g. Xu [2002]; Rydin [2002]; Inkpen and Hirst [2003] and Cimiano et al. [2004]). Her survey is based on several dimensions, of them are two interesting dimensions (besides the dimension of leanring techniques) related to the field of this thesis, the *data source* dimension and the *learning targets* dimension. The data source dimension looks at the world wide web as a source for ontology leanring, arguing that ontology learning is different from ontology aquisition in that it tries to be a fully automated mechanism that looks for sources other than domain experts. This is happening because of the abundance of information on the world wide web, and that workforces for domain knowledge have become increasingly dis-

tributed. In addition, expert knowledge is likely to be incomplete, subjective and even outdated [Zhou, 2007]. Due to this, people have turned to other sources including web documents and database schemas as content for ontology learning, and the web log as a dynamic source. In the learning targets dimensions, concepts, relations and axioms are identified as important targets for the learning algorithm. Each of the learning techniques and methods mentioned focuses on a different aspect of ontology learning and has many variants, making selecting the most appropriate technique difficult.The performance of ontology leanring techniques is usually enhanced with knowledge support of a dictionary or a thesaurus, with borrowed techniques from Natural Language Processing (NLP). Such knowledge provides grounds to calculate similarity between the concepts in statistical methods, and to provide semantic meaning to relations in symbolic methhods.

Ontology learning still has a lot of open issues, including the estimation of co-occurence probabilities not observed in training corpora, and the question of whether more care should be put into designing machine-understandable ontologies or human-understandable ontologies, and how to bridge the gap between these two kinds of ontologies. Other issues include the leanring of specific relations, like part-whole relation which is critical to Sequence Ontology [Eilbeck et al., 2005] for genomic annotation, and the learning of higher degree relations (as opposed to current binary relations) among concepts, and the construction of high-level ontology from fine-grained learning results. Also, the issue of incremental ontology learning, learning from multimedia information other than text and learning from user observations and user queries. For example, if a user combines two terms by "OR" in his query, these terms are probably synonyms [Zhou, 2007].

2.5 Web Recommendation Systems

Since WRS'a utilize web usage mining, the problem of WRS's is an extension to the problem of web usage mining, such that a web log is provided as a collection of user transactions from a certain web server, $\mathcal{W} = \{t_1, t_2, ..., t_m\}$. The mining phase (i.e., Pattern Discovery) can utilize any kind of SPM, clustering or probabilistic modeling of the sequences, to be able to provide the user with recommendations on topics, products, or next page request related to his current pageview and online session.

I consider next page request prediction as the basic form of WRS's, which is based on a probabilistic model, such that a Markov process is used to model the transition between different pageviews (this is discussed in Chapter 6). The common form of WRS's is the top-n WRS. Let us refer to the online user requiring recommendation as the *active user u*. A WRS finds the top-n items that the active user could be most interested in.

For ontology-based WRS's, this thesis assumes the existence of an ontology, based on a domain knowledge, from which inference can be made, relations, and semantic information can be drawn. Let us define ontology-based web recommendation as follows.

Definition 5 (Ontology-based WRS). *Given a sequence database $\mathcal{W}$, and a domain ontology $\mathcal{O}$, define the utility function $\lambda(u, p_i, \mathcal{O})$, which measures the interest of user u in item p_i using relations $\mathcal{R}$, class hierarchies $\leq_{\mathcal{C}}$ and axioms $\mathcal{A}$ from $\mathcal{O}$. An ontology-based WRS finds the top-n items that maximize this function.* $\square$

Only few WRS's use domain knowledge, and those that do, mostly, use it in the form of a topic taxonomy, also called *shallow ontology*. Next, we present a brief survey and an overview of Web Recommendation Systems, and in Chapter 3 we discuss our proposal of ontology-based WRS's.

2.5.1 Overview of Web Recommendation Systems

Web recommendation systems can be divided into the following categories.

1. *Content-based* systems (CB): These systems mine the usage history of the user(s), and are mostly concerned with the web page content, and maybe the semantics surrounding it. Mining methods are used to generate frequent patterns and association rules, which are later used for recommending products and next page accesses. Examples include, WebWatcher [Joachims et al., 1997], SEWeP [Eirinaki et al., 2003], and the works in Spiliopoulou [2000]; Billsus et al. [2002]; Zhang et al. [2002] and Berendt [2002]. In SEWeP keywords are extracted from text surrounding hyperlinks in each web page, and used to cluster pages based on a shallow ontology. Items in association rules resulting from SPM of the web log are mapped to their corresponding clusters to generate recommendations of other items from these clusters. CB suffers from problems of cold start (new items introduced that cannot be associated with existing item clusters), scalability (too many items with too much data about them, e.g., sparse clusters) and content overspecialization (the system recommends items specific to only one topic, i.e., no diversity) [Adomavicius and Tuzhilin, 2005].

2. *Collaborative Filtering* systems (CF): Such system depend on the collaborative effort of users' ratings of a pool of items. Also, the items can be manually or automatically assigned weights without relying on user ratings. Here, clustering and classification mining methods are used to build a system that can provide recommendations on items, to a user based on his peer's ratings (like-minded users). In this case, users are classified into groups or clusters, according to their interest in a subset of the

pages. Users provide ratings to items, which can be represented in a matrix of users-items correlations. Then, the recommender system recommends to the active user items which users in his group are interested in, after mapping the user to his group using semantic similarity measures. Examples of these systems, include WebPersonalizer [Dai and Mobasher, 2003] and the works in Sarwar et al. [2001]; Mobasher et al. [2002, 2004]; Hofmann [2003] and Hofmann [2004]. CF suffers from several problems, most importantly it relies heavily on explicit user input [Li and Zaïane, 2004], and it lacks diversity in recommendations [Anand et al., 2007], the same content overspecialization problem. Baseline algorithms used so far in CF include k-Nearest neighbor, k-Means Clustering, Support Vector Machines *SVM*, and Probabilistic Latent Semantic Analysis *PLSA* [Sandvig et al., 2007]. We notice that CF based on association rule mining or sequential pattern mining is rarely used, and lacks some research.

3. *Hybrid systems*: These combine CF and CB in one system [Melville et al., 2002; Jin et al., 2005; Sandvig et al., 2007]. The idea is that one system will complement the other, in order to overcome its shortcomings. Recently, commercial systems (e.g., Adobe's Omniture Recommendations), provide such hybrid solutions in a comprehensive software application. Disadvantages of such system include the burden put on preprocessing, data collection and data preparation, which make it difficult to update the recommendation model once new data arrives, whether online or offline.

4. Eirinaki and Vazirgiannis [2003] add also *Rule-based filtering*, which is built as an expert system. It asks the user a set of questions from a decision tree, in order to provide a recommendation tailored to his needs.

Top commercial content-based WRS include NetMining[1], which acts as a predictive profiling tool that analyzes web site visitors' browsing behaviors to customize the web site content for each visitor. Redwood[2], is an Open Source web log mining system, implemented with Java, as a web-based interface. More advanced commercial WRS's are mostly provided as CRM (Customer Relationship Management) solutions, that combine usage data mined from the web log, with customer information collected from registration information, browsing behavior, etc., to gather business intelligence, like IBM SurfAid Analytics [3], which helps web managers identify the most effective marketing initiatives, content, and navigation for their web sites. It improves site navigation and page layout,

---

[1]http://www.netmining.com/
[2]http://sourceforge.net/projects/redwood/
[3]http://www.ibm.com/surfaid/

and also provides future application usage prediction, by employing probabilistic Markov models. Omniture Recommendations[1] featured in Adobe Online Marketing Suite recommend products or content such as "people who bought this bought that", most viewed, and top sellers, as well as recommendations based on Omniture SiteSearch queries and results. WebTrends Visitor Data Mart and WebTrends Analytics[2], now used in Microsoft Office SharePoint Services 2007, integrates usage data with content to provide relevant information to web site visitors. As a content-based WRS, ACR News recommends news to the visitors of the web site. It obtains usage profiles by clustering user sessions based on *association rule hypergraph partitioning* [Mobasher et al., 2000]. The recommendations can be made after matching current user session activities and usage profiles by measuring distance and similarity.

On the other hand, commercial systems that use collaborative filtering include GroupLens by NetPerceptions[3], which recommends newsgroup articles based on Pearson-r correlation of other users ratings. Amazon.com [Linden et al., 2003] API, the famous e-commerce web application, enables users to rate products and provide recommendations in the "people who bought this bought that" form, also enabling personalized user browsing experience. It matches recommended products to the products purchased or rated by the customer. Vignette Recommendations[4] determines what content like-minded peers find useful, then displays the best search results, content links, and products. It also provides user's next request recommendations, dynamic content links that are best consumed in a specific order, which reduces clicks to intended content, resulting in increased customer satisfaction.

MovieLens[5] is a CB and CF hybrid WRS for movie recommendations. It combines collaborative filtering and multiple information filtering agents. Weighted feature technique is used for profiling. WebSELL [Cunningham et al., 2001] is another CB and CF hybrid WRS, which recommends products to online shopping customers. Case-based product retrieval technique is utilized, to measure the similarity of the products, to examine whether the products are suitable for the customers' requirements. User profiles are stored in the server as a profile database. The target user is identified in a virtual community by profile similarity matching, so collaborative recommendations can be made by recommending the highly ranked products drawn from the virtual community. Another CB and CF hybrid WRS is LaboUr [Schwab et al., 2001], which maintains a user model by observing users.

---

[1]http://www.omniture.com/
[2]http://www.webtrends.com/
[3]http://www.netperceptions.com
[4]http://www.vignette.com/
[5]http://movielens.umn.edu/

Learning algorithms utilize the model in recommending new relevant objects to the users, and a collaborative approach compares the content-based model to make recommendations based on nearest neighbor and probabilistic techniques.

Major drawbacks of non-ontology-based WRS's, include the *cold start problem* of new users and new items and *sparsity problem*, in addition to other problems discussed in Section 3.1 as part of problems identified by this thesis and the research questions posed.

2.5.2 The Use of Semantics and Content in Recommender Systems

Several attempts have been made to overcome problems in WRS's, by building different models which integrate several sources of information into the different phases of the recommender system. The need for domain knowledge is obvious and important. One can trace the use of domain knowledge in web usage mining to Baumgarten et al. [2000], who defined domain knowledge as *flexible navigation templates that can specify generic navigational behavior of interest, network structures for capturing web site topologies, concept hierarchies and syntactic constraints*. WRS's that rely on Content-Based filtering (CB), combine web content and web structure mining methods. Web content mining allows for the extraction of semantics and keywords from the textual content of web pages, while web structure mining incorporates meta-information included in the hyperlinks and the content surrounding them in an HTML page [Berendt and Spiliopoulou, 2000; Berendt, 2002; Dai and Mobasher, 2003; Li and Zaïane, 2004; Guo et al., 2005; Pabarskaite and Raudys, 2007]. This information helps narrow down the search space and guide the mining algorithm of the WRS. For example, Pabarskaite and Raudys [2007] include text information from HTML link tags registered in the web log with their associated web access sequences, and Berendt [2002] implements a preprocessing module in two systems, STRATDYN and WUM, that classify web log transactions according to concept hierarchies. These hierarchies are built semi-automatically by parsing URLs in the web log, looking for path names indicating entity classes, similar to concepts in a domain ontology. Such classes are presented as constraining templates to the mining algorithm. Li and Zaïane [2004] combine textual content and connectivity information of web pages besides mining the web log, which does not require user input at all. The page textual content is used to pre-process log data to model content coherent visit sub-sessions, which are then used to generate more accurate users' navigational patterns. Structure data of the web site, is used to expand the navigational patterns with a rich relevant content. The connectivity information is also used to compute the importance of pages for the purpose of ranking recommendations.

Web content mining is enhanced if web pages are characterized using an abstract repre-

sentation based on ontologies. This is taken a step further, by incorporating these ontologies into all phases of a WRS, as proposed by this thesis, in Section 3.2.

On the other hand, most CB WRS's employ text mining techniques to extract a set of keywords related to a topic/item presented in a web page. A weight (like TF-IDF [Salton and Buckley, 1988]) may be associated with these keywords, which are later clustered using similarity measures. The goal is to create semantic profiles for items such that recommendations can be expanded to include relevant items from these profiles. In WebPersonalizer [Dai and Mobasher, 2003], each pageview is represented as a k-dimensional feature vector, where k is the number of features, showing the weight of each feature in the given pageview. These features represent keywords extracted from the pageviews. Then for the whole collection of pageviews, a matrix of feature weights is collected. For recommendation, the active user profile, is matched against a set of aggregate usage profiles, showing which group of users the active user shares common interests with, then recommendations are provided from the set of items common among these users. Dai and Mobasher [2003] argue that such content features can be integrated only into pre-mining or post-mining phases of web personalization. The disadvantage of this approach is its high dimensionality and its incapability of capturing more complex relations among objects, since only a shallow ontology is used. The difference between keywords extraction and content features extracted from web pages, and the use of semantic objects drawn from a domain ontology, is that the later can model object properties and relationships based on the underlying domain ontology, while the other ones cannot. Being able to model objects' properties and the granular relationships that connect these objects, provides the ability for the recommender system to generate its recommendations, as direct instantiations of these objects. Also, this kind of modeling enables more accurate calculations of similarity measures, since more detailed relations are involved.

Another example is that of Quickstep and Foxtrot [Middleton et al., 2009] research paper recommenders, in which an external shallow ontology containing only *is-a* relations is used in user profiling. Each user's known publications are correlated with the recommender system's classified paper database. A set of historical interests is compiled for the active user, and the relationship analysis tool provides a ranked list of similar users. Example of other WRS's that depend on subject taxonomies as shallow ontologies to classify web pages include Cooley et al. [1999]; Acharyya and Ghosh [2003]; Eirinaki et al. [2003]; Middleton et al. [2004] and Ziegler et al. [2005]. Several subject taxonomies used, are publicly available, like Yahoo! Directory[1], Lycos[2], and ODP. In these systems, pageviews

---

[1]http://dir.yahoo.com/
[2]http://yellowpages.lycos.com/

from the web log, are mapped to topics in the taxonomy, so that association rules and frequent patterns, resulting from mining, can be expanded to include other items from these taxonomies as future recommendations to the user. Ziegler et al. [2005] propose the use of light taxonomies (like Amazon.com book taxonomy) to represent users as vectors of scores assigned to topics rather than individual items. Interest shown by a user in a specific item propagates up the taxonomy to all supertopics, for generalization purposes.

On using a domain ontology, Oberle et al. [2003] present the SEAL framework for enhancing web usage transactions with formal semantics based on an ontology underlying a web portal application, and used to annotate its resources. They tackle the problem of mapping many semantic objects to a single web page, but do not elaborate on how it is done, or discuss its consequences. The presented system resembles a combination of WebPersonalizer and WUM [Berendt and Spiliopoulou, 2000]. After obtaining the Semantic Log File, and converting it into *arff* format from the WEKA system [Witten and Frank, 1999], clustering is employed for mining this semantic web log, to extract knowledge about groups of users, users' preferences, and rules. Since the proposed framework is built on an ontology, the web content is inherently semantically annotated exploiting the portal's inherent RDF annotations. The authors show interesting results on concept generalization, that is made possible by the semantic representation in the Semantic Log File. On the other hand, they focus only on mining, and when association rules mining is applied, it resulted only in trivial rules, not up to expectations, given the semantic-rich log.

Tags[1] are used in the area of social networking [Hayes et al., 2007] to cluster bloggers and posts. Diederich and Iofciu [2006] study how users create tag profiles corresponding to their interests and receive recommendations based on user-user similarity built using the tag profiles in a user-based CF algorithm. Similarily, Zanardi and Capra [2008] use cosine to compute user-user and tag-tag similarities to propose a social ranking formula for research papers recommendation. They also propose a query expansion method that does not consider semantic relations among concepts in a fashion similar to kNN clustering [Zanardi and Capra, 2008]. de Gemmis et al. [2008] rely purely on WordNet such that documents are mapped to synsets to identify semantic concepts behind them. This is augmented with a probabilistic model for learning user profiles. User tags are treated as additional content in documents. This approach is applied on a cultural hertiage recommender, but might not work for e-commerce applications where semantic relations extend beyond lingual semantics. In the approach proposed by our thesis, we rely on an expert domain ontology from which concepts are identified.

On the other hand, Niwa et al. [2006] propose a cluster-based algorithm for recommend-

---

[1]Keywords that users attach to items in the web site.

ing web pages based on the pages users have tagged. The recommendation is straightforward and is based on the similarity of $tf \cdot idf$ tag profile vectors. A similar approach [Cantador et al., 2010] emerged at the time of preparing this thesis. In this approach, no ontology is used, and tags are used directly in $tf \cdot idf$ measures to compute weights in user and item profiles without dimensionality reduction or clustering, although such an approach can be applied in any scenario where users tag items, but cases of large numbers of tags will pose a scalability issue in these approaches.

In Guan et al. [2010] user provided tags are used for document recommendation, while a domain ontology is not utilized, relations between users, documents and tags are found by constructing weighted bipartite graphs, and an algorithm for subspace leanring to find the optimal semantic space. The reason claimed is that straightforward vector space representation does not consider semantic correlations between tags, which is important for recommendation. Our approach on the other hand, does consider such correlations since tags belong to ontological concepts with relations of several kinds among them in the ontology. As we propose a novel method for dimensionality reduction, Guan et al. learning algorithm still suffers from a scalabiltiy problem.

Our proposed ontology-based recommender system in Section 7.2 can be described as content-based, and it differs from the discussed methods in that, it relies on user-provided tags only, without the need to retrieve keywords from web documents, or the need for rating information. It also relies on a provided domain ontology to which the tags are mapped, without using clustering or annotating the items with ontology concepts.

2.5.3   Discussion

Web recommendation systems have come a long way since they were first introduced in the late 1990s. They went from simple results of data mining to complete stand-alone applications that encapsulate data mining in their process. Companies like IBM and Adobe, now provide WRS's as complete CRM solutions. We have seen how similarity metrics have been used to group similar items, and to group like-minded users, forming different matrices and applying different statistical methods. We have also seen how web usage mining is used in content-based WRS, when the web log is enriched with semantics in the form of categorical information or keywords. The followings are major achievements that can be noted in the field of WRS's in the last decade:

1. WRS's have led to semantic enrichment of web logs and user transactions, e.g., *C-Log* in SEWeP [Eirinaki et al., 2003], and Semantic Log in SEAL [Oberle et al., 2003].

2. WRS have utilized statistical modeling of user profiles, e.g. vector space representation of user transactions in WebPersonalizer [Dai and Mobasher, 2003].

3. Sequential Pattern Mining for WRS's has been enhanced, in different ways.

    (a) Input is being filtered, by using templates based on results of web structure mining and web content mining [Baumgarten et al., 2000; Berendt and Spiliopoulou, 2000].

    (b) The introduction of faster algorithms with less memory consumption, like PrefixSpan [Pei et al., 2001] and PLWAP [Ezeife and Lu, 2005].

    (c) Enhancing precision and recall in probabilistic mining methods, with the introduction of Selective Markov Models [Deshpande and Karypis, 2004], and the use of Support Vector Machines and Probabilistic Latent Semantic Analysis [Jin et al., 2004].

4. WRS utilize a categorization, or a topic taxonomy, to explain recommendation results and provide generalized recommendations, like semantic recommendations and concept recommendations in SEWeP [Eirinaki et al., 2003], but do not yet make full use of semantic web technologies and domain ontologies to extend recommendations and solve problems, like content overspecialization.

Web recommendation systems have made a lot of contributions in the way we use and think of the Internet. The functional achievements of WRS include services and applications, in the areas of research publication recommendations (e.g., DBLP[1]), news article recommendations (e.g., ACR News), movie title recommendations (e.g., MovieLens[2], NetFlix[3]), and of course important contributions in the area of e-commerce and online shopping (e.g., Amazon.com, eBay). WRS's are now an integral part of many CRM tools (like IBM SurfAid Analytics), especially with the advent of social networks, like Facebook[4]. They provide more user-friendly interfaces and personalized experiences for users, thus increasing customer retention and loyalty.

---

[1]http://www.informatik.uni-trier.de/ ley/db/
[2]http://movielens.umn.edu/
[3]http://www.netflix.com/
[4]http://www.facebook.com

Chapter 3

Proposed *SemAware* System for Web Recommendation

The surveys presented in Chapter 2 reveal a number of problems and pitfalls associated with Sequential Pattern Mining and Web Recommendation algorithms. In this Chapter we discuss these problems and identify the questions that drive the research in this dissertation. Then, a proposed solution is discussed and summarized before providing details in subsequent chapters. In addition to the research questions proposed here, thesis contribution is discussed in Section 1.2.

3.1   Problems Identified and Research Questions

Through the surveys conducted on WRS's and SPM in Chapter 2, the following problems are identified:

1. SPM algorithms have a huge and a sparse search space when applied on web log mining. A method is required to efficiently guide the algorithm in this search space.

2. Mathematical modeling of sequences in SPM is needed to enable n-item prediction and sequence re-generation. This can be a polynomial or probabilistic model capable of re-generating the sequence with a minimal level of error.

3. Web Recommendation Systems suffer from the following problems:

    3.1 The **Cold Start** problem, which appears when a new item is introduced that was not recommended before. This new item did not surface up in the usage history, or is not rated by users yet, so the system is not able to recommend it,

since the process relies on existing user transactions [Li and Zaïane, 2004]. The same problem appears when a new user approaches a content-based WRS with no previous browsing history, the system has no way of telling what the next page request will be, or what proper item to recommend. The system requires some time to pick up user's usage information.

3.2 The problem of **Sparsity and Scalability** affects efficiency when there is a huge number of items and weights/ratings associated with each item and when a profile is maintained for each user. This problem leads to large matrices of data that require transformation and processing on the fly for online recommendations. Sparsity problem appears when there is a huge matrix with only few weights or ratings. Differently speaking, this problem arises when there are many items to be recommended, but only few recommendations are provided, or recommendations are mostly targeting only a subset of the items. The same problem appears in SPM when dense data sets are mined, requiring an effective pruning method to reduce the search space.

3.3 **Content Overspecialization** occurs in such systems as they recommend items of high relevance to the user, with no diversity in results. This problem affects both the user and the system. The user may get bored fast and not come back. The system may not be able to introduce new items.

3.4 The **Gray Sheep** problem occurs when the active user has a taste that is not similar to any group of users, or a profile that has equal similarity over most item profiles. In this case recommendation with association rules only is not a good idea since these rules depend on other users' browsing behavior and not only on this single active user's browsing history. It cannot be said that every user is a gray sheep, only users that the system find very difficult to label as belonging to one certain cluster of users.

3.5 WRS's do not have support for **Multi-Criteria Matching**. A similarity measure that considers different criteria is definitely needed for user-item matching, since a user might have different interest levels in an item depending on different criteria, like quality, durability, price...etc.

In light of the problems identfied, we put forth the following research questions to be answered by this dissertation.

Research Question 1 Can semantic information drawn from a domain ontology enhance Sequential Pattern Mining algorithms and Web Recommendation Systems over the

use of a simple topic taxonomy?

Answer: The answer to this question lies in the difference that a domain ontology has over a topic taxonomy, which is the rich collection of granular relations $\mathcal{R}$ (e.g., *has-a*, *part-of*, *requires*), relation hierarchies $\leq_{\mathcal{R}}$ that act as weights for these relations, and axioms $\mathcal{A}$. These relations provide an accurate measure for semantic similarity between items. Using content semantic similarity depicted only by categorization or *is-a* taxonomic relations is not enough. In Chapter 4 we propose a semantic similarity measure that incorporates the different relations of an ontology. In Chapter 5 we show how these relations are used in SPM to reduce the search space and guide the mining algorithm. In Chapter 6 they are used also to solve the contradicting predictions problem and prune the search space of Markov models, and in Chapter 7 we illustrate their use to expand the recommendation set.

Research Question 2 Is there a way to integrate semantic information in all phases of Web Usage Mining?

Answer: Yes, to answer this question we propose SemAware in Section 3.2, a comprehensive web usage mining and recommendation system in which semantic information drawn from the domain ontology is integrated in the three phases of Pre-processing, Pattern Discovey and Post-processing. Subsequent chapters in this thesis are dedicated to discussing the integration at each of these phases.

Research Question 3 Can the inclusion of semantic information solve the problems identified in WRS's and how?

Answer: The methods outlined in this thesis use domain ontology to solve these problems as follows:

For the *cold start* problem, when a new item arrives at the web site and into the domain ontology, properties of this item are used to find its relations to other concepts in the ontology, and that guarantees its appearance in a recommendation set of other similar items, using cosine similarity. When a new user arrives at the system with no browsing history, tags provided in the user query are mapped to ontology concepts and used to find similar items for recommendation using SemAwareIN algorithm, based on a vector-space model of item-concept correlations, as proposed in Section 7.2.2. To provide a scalable recommendation system and solve the *sparsity* problem, a dimensionality reduction method (SemAwareIN-LCA) is proposed in this thesis, that relies on the domain ontology to combine leaf concepts with their Lowest Common Ancestor (LCA) and reduce the item-concept correlation matrix. As for the problem of *content overspecialization*, Spreading Activation is used in SemAwareIN-

Ex algorithm during post-processing to traverse the domain ontology and expand the recommendation set with diverse recommendations, relying on ontological relations among the recommended items. As for the *gray sheep* problem, user-provided tags are matched against ontology concepts via a thesaurus to generate an appropriate set of items for recommendation using cosine similarity (Section 7.2.2). In Collaborative Filtering and Hybrid models, we suggest that user profiles be projected over clusters of ontology concepts, such that user similarities can be found at different settings with different clusters. This means the ability to find that a user is more similar to a certain group of users for a certain subset of the ontology concepts.

Research Question 4 What impact does the inclusion of semantic information have on recommendation results?

Answer: The ontology-based model is richer and better defined than a keyword or an item-based model. It provides grounds for the representation of fine-grained user interests (e.g., interest for items such as an actor or a camera part). The ontology provides a more formal processable meaning to concepts (such as what an underwater camera requires, what movies a certain director made) and it supports inference mechanisms to enhance recommendations. In Section 7.3 we show how a proposed Spreading Activation method over the ontology increases the accuracy of a WRS, and in Chapter 5 we show how semantic information can minimize the tradeoff between complexity and accuracy in Markov models.

3.2 Proposed System and Solutions - SemAware

We propose the use of a domain ontology, and its integration in a complete web usage mining system, targetting web recommendation and web usage mining.

Thesis hypothesis: The addition of domain ontology components (concepts $\mathcal{C}$, concept hierarchy $\leq_{\mathcal{C}}$, relations $\mathcal{R}$, relation heirarchy $\leq_{\mathcal{R}}$, and axioms $\mathcal{A}$) to Sequential Pattern Mining and into the utility function $\lambda(u, p_i)$ of user-item interests in WRS's to become $\lambda(u, p_i, \mathcal{O})$ is possible, and will increase effectiveness of these systems by solving the problems of cold start, scalability and sparsity, content overspecialization, contradicting predictions and complexity-accuracy tradeoff. $\square$

To validate this hypothesis we propose *SemAware*, a three phase framework for ontology-based web recommendation and web usage mining. Experimental results at each phase of SemAware show the increased effectiveness as claimed.

Figure 3.1 shows all the components of *SemAware*. The figure shows the three phases,
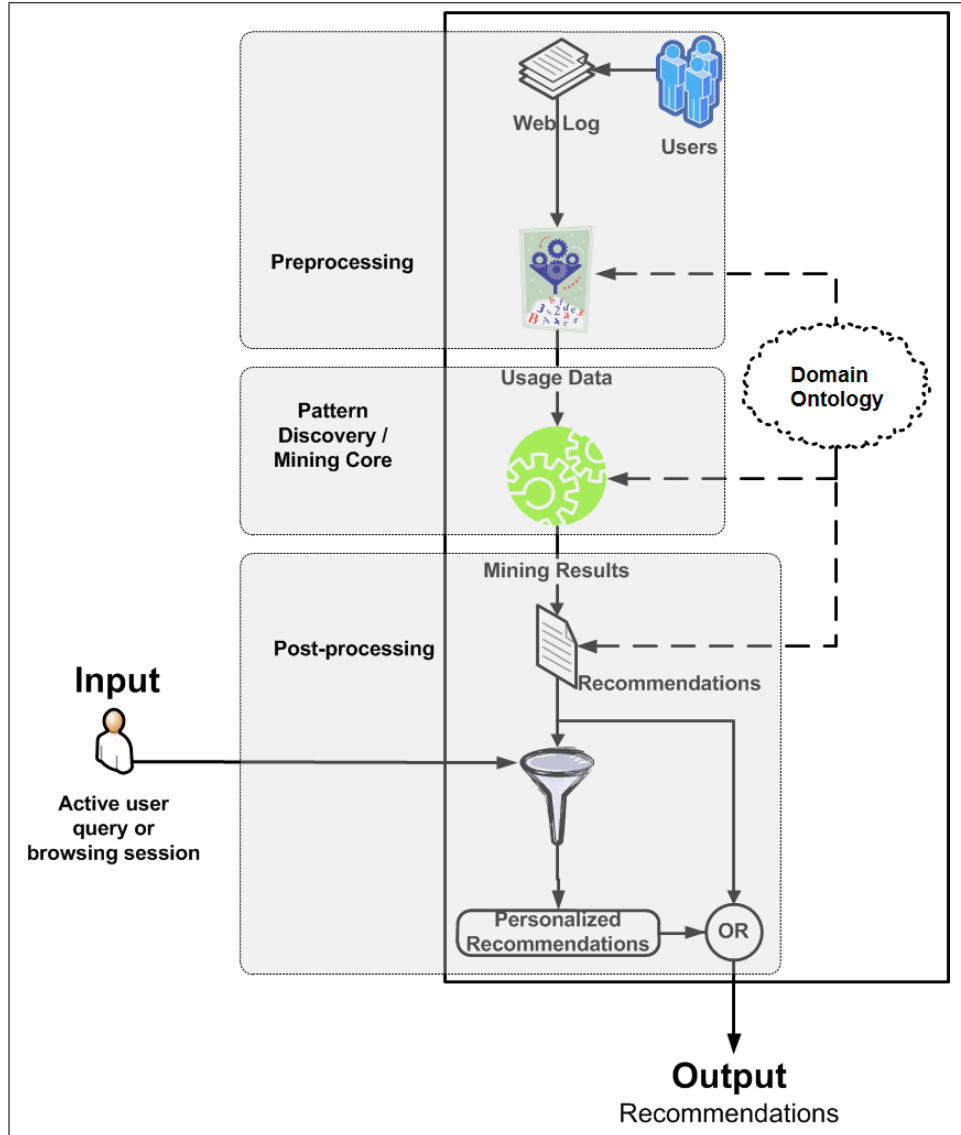
Figure 3.1: SemAware framework, built on top of the three phases of web usage mining.

namely, preprocessing, pattern discovery, and post-processing (from top to bottom). Here we provide an overview of SemAware, then each subsequent chapter discusses a separate phase of this proposed system.

In the first phase, a clean server-side web log is provided for preprocessing. This log contains all information about user access patterns. Items and pageviews p_i in the web log are mapped to their corresponding ontology concepts turning the web log into a sequence

database $\mathcal{W}$ of semantic objects (shown in the figure as "usage data"), thus enriching the web log with semantic information.

The second phase is Pattern Discovery and Mining. The semantic-rich sequence database $\mathcal{W}$, is fed into this process. Semantics-aware SPM algorithms, *AprioriAll-sem*, *GSP-sem*, and *PLWAP-sem* are proposed in which the semantic information from the first phase is used to prune candidate sequences and avoid support counting when the candidate sequence is not semantically correct. These proposed algorithms demonstrate, with experimental results, that domain ontology can be used in the mining core of any SPM algorithm, and it enhances its perfromance and effectiveness, without compromising the quality of the generated frequent patterns under some conditions. We take this a step forward and propose to integrate this semantic infomation into a probabilistic model for prediction, namely a semantics-aware Selective Markov Model, using Markov processes as opposed to SPM, for page prefetching. The idea is to show that this integration minimizes the tradeoff problem between accuracy and complexity in these Markov models, at the same time it solves the cold start and the contradicting prediction problems in these models.

In addition to next page prediction, results of the second phase include the frequent patterns found by semantics-aware SPM, these are input to the third phase that uses them to generate semantics-rich association rules and top-n recommendations. This post-processing phase includes techniques that allow further filtering and reporting of results. The interpretation of these results represents them as a set of recommended items to the active user, or as decision making recommendations to the administrator, to be used in web site restructuring or marketing campaigns. The actual operation of the WRS lies in this phase. As the active user presents her query, or requests a pageview, the WRS uses the results of mining to provide recommendations related to her actions. SemAware utilizes semantic knowledge in this phase to provide an interpretation that can be easily understood by a naïve user. For example a pattern that shows two sequentially frequent pageviews like $< p_5 p_3 >$, can be better interpreted if the semantics of both web pages are involved, to result in something like, "*buying a SLR camera is always associated with buying a zoom lens*", from which this interpretation can be inferred "*people who buy a camera will most probably buy a lens to go with it,*" using ontology concept generalization. We focus on displaying the accuracy of these association rules by showing how they provide an informed recommendation when confidence measures fail to do so, by combining semantic-rich association rules with semantics-aware Markov models. In the third phase SemAware provides the ability to generate top-n recommendations. In this case, an item-concept matrix that is populated with each item's similarity to each ontology concept, is used to find the top-n items that maximize the utility function λ. The goal is to show how

Ontology-based recommendation can make use of the recent Web 2.0 tagging technologies to provide top-n recommendations from user-provided tags, with no semantic annotation of the items, solving the problems of cold start, sparisity and scalability, and content overspecialization.

Chapter 4

Preprocessing and Semantic Enriching of the Web Log

Web usage mining assumes the existence of a clean web log in the form of a sequence database of user access transactions as defined in Definition 1, (page 10). This chapter shows how the web log can be enriched with semantic information drawn from the underlying domain ontology. We discuss cases in which the items are annotated and cases in which the items are not annotated with ontological concepts, due to lack of ontology. The goal of this pre-processing phase in SemAware is to provide the mining algorithm with semantic-rich user access sequences.

4.1 Preprocessing in SemAware

In Section 1.1 we presented eMart, an e-commerce application that sells electronic devices and accessories, like Cameras and Photography equipment. Consider Table 4.1 as a sample of a sequence database from eMart. We will adopt this sequence database for all examples throughout this thesis. We assume that user browsing is organized in the clean web log (e.g., Table 4.1) into sessions (also called transactions) denoted as t_i. Sessionizing was discussed in Section 2.2. We avoid cases of tabbed browsing by which a single user might open more than one web page simultaneously in several tabs in his web browser tool.

Web pages (e.g., p_1 from Table 4.1) residing on eMart's web server are annotated with semantic information. In general, a web page can be annotated with semantic information using automatic or semi-automatic software tools, that enable the ontology engineer to extract information from the ontology and use an XML-based language to annotate a web page by showing what semantic information it carries, and where in the ontology it

Table 4.1: Example Sequence Database from eMart.

Trans ID	Access Sequence
t_1	$p_2p_3p_2p_1p_5$
t_2	$p_2p_1p_3p_2p_1p_5$
t_3	$p_1p_2p_5$
t_4	$p_1p_2p_5p_2p_4$
t_5	$p_1p_2p_1p_4$

belongs, some software tools are OntoMat Annotizer [Handschuh and Staab, 2002], MnM [Vargas-Vera et al., 2002], WebOnto [Domingue, 1998], and SHOE [Luke et al., 1997]. This annotation process takes place during the design of the e-commerce application. We use OntoMat Annotizer[1] to manually annotate pages in eMart. Each pageview p_i is mapped to a concept $c_k \in \mathcal{C}$ in the provided ontology. Items dealt with in the mining process are instances of these concepts (also called classes). For example, a "Canon PowerShot A2000 IS" described in web page p_2 (Figure 4.2) as a brand of a digital still camera sold on *eMart*, is an instance of the *Digital* class which is a subclass of *Still Camera* class. The subclass relation is represented by the *is-a* edge in the presented ontology of Figure 2.5.

4.1.1 Lack of Ontology and Ontology Crafting from the Web Log

In the absence of a domain ontology and annotation, SemAware has *OntoCraft* (Algorithm 1) to extract the keyword(s) from each web page p_i's <meta> tag, which holds keyword(s) of topics or concepts concerning the page content. An example is the following HTML line from a web page p_i which describes a camera lens product.

<meta name="keywords" content="lens,camera,optical">

These tags can also be user-provided tags. The HTML line above shows a set of keywords $K^{p_i} = \{lens, camera, optical\}$, such that $k_1^{p_i} = \{lens\}$, $k_2^{p_i} = \{camera\}$, and so on. For every keyword $k_j^{p_i}$, if it is not already in the ontology (an empty ontology includes only the *thing* root node), it will be added as a separate concept in the ontology, and an *is-a* edge is added to connect this new concept with the all concepts from $K^{p_{i-1}}$. This process is repeated for all the pages represented by pageviews in the web log. A detailed procedure is provided in Algorithm 1, with the following example.

Example 4.1 (ontology crafting): Consider the transaction $t_1 = \{p_2, p_3, p_2, p_1, p_5\}$, from the running example in Table 4.1. Algorithm 1 starts by creating the root class *thing* in the ontology $\mathcal{O}$ (lines 1-2) as in Figure 4.1a. Next, the algorithm will search

---

[1]http://annotation.semanticweb.org/ontomat/index.html

Algorithm 1 *OntoCraft()* procuedre for crafting a basic ontology for SemAware from pageview sequences in the web log

---

Input: clean web log $W=\{p_1, p_2, \ldots, p_i, \ldots, p_l\}$, such that p_i represents a single pageview.
Output: Domain Ontology, represented as the directed graph $\mathcal{O} = (V, E)$, such that V is the set of ontology concepts, and E is the set of *is-a* relation arcs connecting the concepts.
Algorithm:

1: Create an empty ontology with only *root* class $V = \{c_0\}$
2: Initialize a counter $j = 0$ for classes that will be added to the ontology
3: /*process each page individually*/
4: **for** i=1 to l number of pages in web log W **do**
5:    Search for $< meta\ name\ =\ "keywords"\ content\ =\ "keyword_1,\ keyword_2,\ \ldots,\ keyword_n" >$ tag in HTML code of web page p_i
6:    /*extract keywords from the HTML tag*/
7:    **for** k=1 to n, number of keywords in page p_i **do**
8:       Increment classes counter, $j = j + 1$
9:       **if** $keyword_k$ was not encountered before **then**
10:          create a class c_j and add it to ontology $\mathcal{O}$, $V \leftarrow V \cup c_j$
11:       **end if**
12:       create an *is-a* arc connecting this class to all previous (now parent) classes created from p_{i-1}, call them C^-, such that a class cannot link to itself
13:       add this edge to E, $E \leftarrow E \cup arc(c_j, c_x), c_x \in C^-)$
14:    **end for**
15: **end for**
16: **return** Ontology $\mathcal{O} = (V, E)$
   end

---

the HTML code of the first page in t_1, which is p_2 (shown in Figure 4.2), looking for the <meta name="keywords" ...> tag which is usually found in the <head> section of HTML code (lines 3-6). In this example, the following line is retrieved <meta name="keywords" content="camera, digital, still, photo">.   In lines 7-13, the algorithm next creates a class c_j for each keyword extracted, and creates an *is-a* relation connecting each new class to its parent class, in this case the *thing* class, as shown in Figure 4.1b. In the second iteration of line 4, the algorithm will fetch page p_3 from t_1, with the set of keywords <meta name="keywords" content="memory-card, storage">, and creates two new ontology classes for these keywords, then creates an *is-a* relation connecting each new class to previous parent classes created from processing p_2, as shown in Figure 4.1c. In the third iteration, the accessed page is p_2 again, by executing code lines 9-11, nothing is added to the ontology. In the next iteration of line 4, the algorithm will fetch page p_1 from t_1, with keywords <meta name="keywords" content="storage,camera-case">, only one new class is created for keyword "camera-case", *is-a* relations connect this class with parent
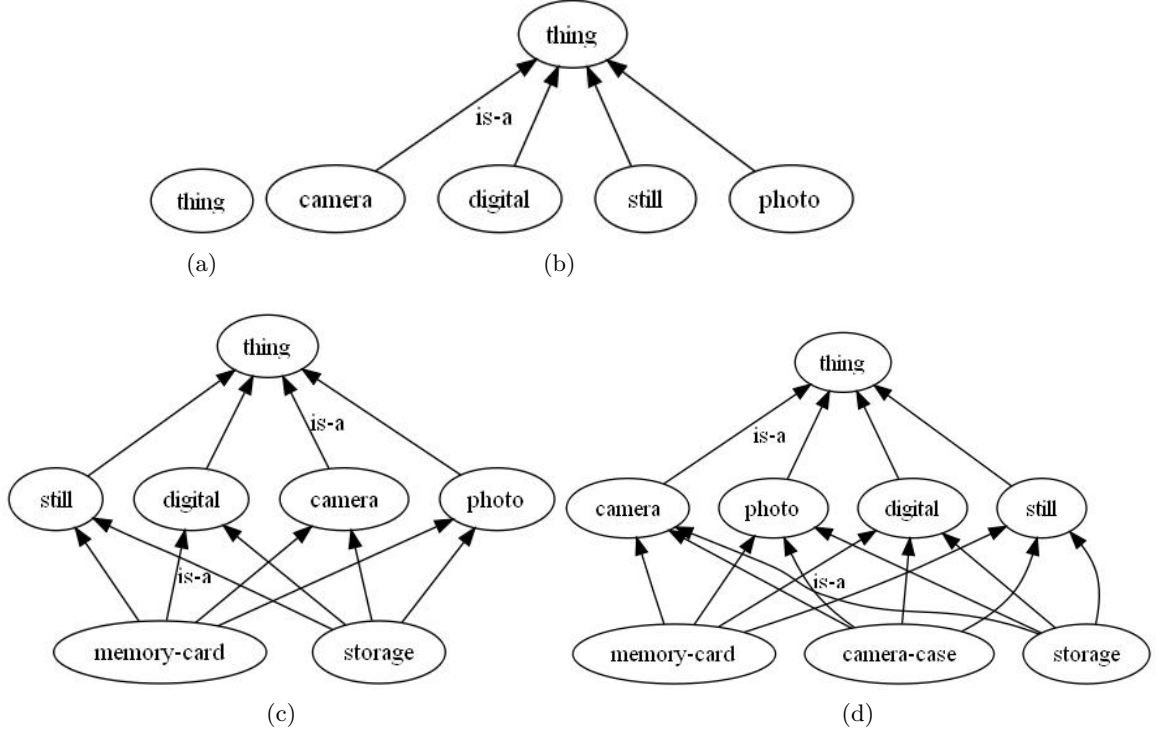
Figure 4.1: Different stages for building a simple ontology from a web log.

classes represented by p_2 as it is the preceding page to p_1 in the clickstream, as shown in Figure 4.1d. Finally, the last page in t_1 is p_5, with keywords <meta name="keywords" content="memory-card">, and by executing code lines 9-11, nothing is added to the ontology. The final crafted ontology is shown in Figure 4.1d.

OntoCraft is a naïve algorithm that treats keywords as nouns and considers each one as an ontology concept, thus the high number of interconnecting relations. There is still a lot of future work to be done to enhance this algorithm to the level of an ontology learning algorithm. Things to consider include the fact that several keywords can refer to one concept. The use of clustering is important to group similar keywords into a single concept. Then a method is required to label the concept and other methods to find the relations between these concepts. We believe that frequent patterns (resulting from regular SPM of the web log) can be used to find relations between concepts. More frequent patterns that reflect the same relation mean that the relation is of high importance to be included in the ontology, less frequent patterns mean that the relation can be ignored and not included in the ontology. The same way can be used to assign relation hierarchy (i.e., a weight) to each learned relation. It is far more difficult to learn granular relations than to learn the

47
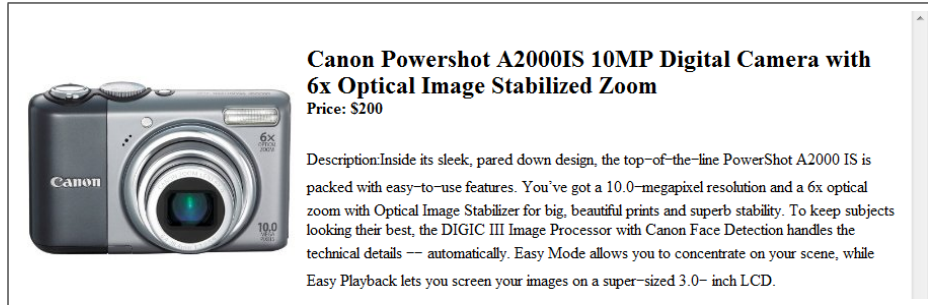
(a) web page p_2

```
<html>
<head>
<title>eMart</title>
<meta name="keywords" content="camera,digital,still,photo">
<meta name="Author" content="Nizar Mabroukeh 2010">
<meta name="Address" content="mabroukeh at yahoo dotcom">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1256">
</head>
```

(b) Header part of the HTML code for p_2

Figure 4.2: Visual rendering and HTML code of pageview p_2 in the running example.

simple is-a relation among the concepts. Most of the work in this thesis assumes that a rich domain ontology is provided as defined in Definition 3, page 24.

4.1.2 Converting Pageviews into Semantic Objects During Preprocessing

Each item in the web log, representing a web page access, is converted into a semantic object in SemAware using Algorithm 2 (page 54). The result of this process is semantic-rich user transactions, which are sequences of semantic objects.

Definition 6 (Semantic Object). *A semantic object o_i is represented as the tuple $< pg_i, ins_k >$, where pg represents the web page containing the product, usually an URL address of the page, and ins_k is an instance of a concept $c_k \in \mathcal{C}$, from the provided ontology $\mathcal{O}$, that represents the product being referenced, where i is an index for an enumeration of the objects in the sequences in $\mathcal{W}$.* □

Sometimes a single pageview p_i may contain more than one semantic object, like a catalog page of products. This presents the problem of mapping many semantic objects to one or many concepts in the ontology. Depending on the number of objects in p_i, there can be different mapping scenarios, as follows (also illustrated by Exmaple 4.2 below):

1. If a given p_i contains only one semantic object, then mapping is directly provided as **one-to-one**, resulting in $o_i =< pg_i, ins_k >$.

2. In the case of p_i referring to many concepts, many-to-one or many-to-many mapping can be made. A **many-to-one** mapping is made by crawling up the concept hierarchy $\leq_{\mathfrak{C}}$ until a common ancestor of the instances in p_i is reached, to which p_i will be mapped.

3. In the case of **many-to-many** mapping, each instance in p_i is mapped to a single concept, and treated as a single semantic object o_{i_k}, such that $o_{i_k} =< pg_i, ins_k >$ and $o_i = \bigcup_k o_{i_k}$.

4. There is no **one-to-many** mapping, because SPM will be carried on the sequences of semantic objects, becaue an item is mapped to only one semantic concept.

Eventually, the web log $\mathcal{W}$ is converted into a sequence of semantic objects $\mathcal{W} = \{o_1, o_2, \ldots, o_m\}$, where $m \geq l$, called semantic-rich user transactions.

Example 4.2 (object mapping): This example shows how pageviews in transaction $t_9 = \{p_5, p_4\}$ are mapped to the ontology and converted into semantic objects, as described in lines 9-29 of Algorithm 2 (page 54). The first web page in t_9 is p_5, according to lines 9-10, the algorithm will fetch tagging keywords from the HTML code of p_5, as <meta name="keywords" content="memory-card">, in this case one keyword, and add it to list $K^{p_5} = \{memory - card\}$. Next, by executing the conditional statement of line 12, the algorithm will map this keyword to the ontology class $memory - card$ in Figure 4.1d as one-to-one mapping, by executing lines 5-6, in which a semantic object is created as the tuple $o_5 =< $"elec/sd.htm"$, c_5 >$, where c_5 is the $memory - card$ class in the ontology (with serial number 5). o_5 is then added to the list of semantic objects, $so = \{o_5\}$. Next in the transaction, is p_4, with the keywords <meta name="keywords" content="memory-card, digital">, the algorithm is faced with more than one keyword, here is the option of using many-to-one mapping or many-to-many mapping. In the case of many-to-one mapping (code lines 17-21), $K^{p_4} = \{memory - card, \ digital\}$, the algorithm will climb up the class hierarchy of the ontology to find a common ancestor between the two classes of $memory - card$ and $digital$, which is the root class $thing$ as seen in Figure 4.1d. Next, a semantic object is created as the tuple $o_4 =< $"elec/sandisk.htm"$, c_0 >$, where c_0 is the $thing$ class in the ontology. The object o_4 is then added to the list of semantic objects $so = \{o_5, o_4\}$. If many-to-many mapping was chosen (code lines 23-27), then every keyword in K^{p_4} is mapped separately to the ontology resulting with the semantic objects

$o_{4_1} =<$ "*elec/sandisk.htm*", $c_5 >$ and $o_{4_2} =<$ "*elec/sandisk.htm*", $c_3 >$, which are later added to the set of semantic objects *so* as $so = \{o_5, < o_{4_1}, o_{4_2} >\}$.

At the end the sequence database from Table 4.1 is transformed into a semantic-rich sequence database as in Table 4.2, by mapping each pageview to its correponding ontology concept and generating semantic objects after consulting Table 4.3, where each semantic object entry o_i is made of the tuple $< pg_i, c_k >$, such that $c_k \in \mathcal{C}$. Table 4.3 is constructed for the running example as a look-up table to show mappings of pageviews, in reality, Algorithm 2 is used as illustrated in Example 4.2 above.

Table 4.2: Resulting semantic-rich sequence database after preprocessing of the web log, using Algorithms 1 and 2.

Trans ID	Access Sequence
t_1	$o_2 o_3 o_2 o_1 o_5$
t_2	$o_2 o_1 o_3 o_2 o_1 o_5$
t_3	$o_1 o_2 o_5$
t_4	$o_1 o_2 o_5 o_2 o_4$
t_5	$o_1 o_2 o_1 o_4$

Table 4.3: Assumed domain knowledge contained in each accessed pageview, last column shows generated semantic objects.

Access-ed Page	Actual corresponding web page	Ontology concept	Semantic Object
p_1	/cameras.html	*Cameras*	o_1
p_2	/cameras/canon.html	*Still Camera*	o_2
p_3	/chem/fsoln.html	*Film developing solution*	o_3
p_4	/film/videofilm.html	*Video Film*	o_4
p_5	/elect/dbatteries.html	*Dry Battery*	o_5

4.2 Computing The Semantic Distance

Besides semantic enrichment of the web log, the preprocessing phase of SemAware also computes the semantic distance between each semantic object and each other semantic object in the web log. This semantic information plays an important role in the mining phase of SemAware to produce effective semantics-aware SPM algorithms and in next page request prediction, as will be seen in Chapters 5 and 6.

Definition 7 (Semantic Distance). *The Semantic Distance γ_{o_i,o_j} is a measure of the distance in the ontology $\mathcal{O}$ between the two concepts of which o_i and o_j are instances.* □

Definition 8 (Semantic Distance Matrix). *A Semantic Distance Matrix $\mathcal{M}_{n \times n} = \{\gamma_{o_i o_j}\}$ is a $n \times n$ matrix of all the semantic distances between the n semantic objects represented by pageviews in the sequence database $\mathcal{W}$.* □

The semantic distance matrix $\mathcal{M}$ is not necessarily symmetric, as the semantic distance between two ontology concepts (e.g., *Digital Camera* and *Batteries*) is not always the same from both directions, espcially when relations are weighted. For example, the relation between the concepts of *Camera* and *Battery* is not symmetric, as one would assume that a camera requires a battery to operate, and users who buy cameras will probably buy some batteries later on, but when people buy batteries it does not necessarily mean that they own a camera. One can say that the semantic distance is the measure in units of semantic relatedness between any two objects o_i and o_j. The more related two objects are, the lower is their semantic distance. For computing semantic distance, this thesis follows an intensional approach by exploiting features defined directly over concepts. The straighforward and direct way is to use graph-based intensional approach by measuring the distance as the number of separating is-a edges between the concepts that represent o_i and o_j, e.g., using Wu and Palmer measure [Wu and Palmer, 1994]. But this approach does not capture all the semantic relations among the concepts that a rich domain ontology provides. A more meaningful measure is to compare the relations that the two concepts posses, and compute their similarity based on the idea that two concepts can be considered similar if the other concepts to which each of them is connected are the same. For this we resort to set theory and adopt the Jaccard Coefficient which measures the similarity between two sets [Jaccard, 1901]. We propose the following semantic distance measure using Jaccard Coefficient:

$$\gamma_{o_i,o_j} = 1 / \frac{\left| \mathcal{C}_{c_i}^- \cap \mathcal{C}_{c_j}^- \right|}{\left| \mathcal{C}_{c_i}^- \cup \mathcal{C}_{c_j}^- \right|} \tag{4.1}$$

such that the semantic distance is the inverse of the semantic similarity between the two objects. The semantic similarity depends on the Jaccard Coefficient that measures the similarity between two sets, where the set $\mathcal{C}_{c_i}^-$ is the set of all concepts that are directly connected (via any one relation $r \in \mathcal{R}$) to the concept c_i to which the object o_i belongs, $\mathcal{C}^- \subseteq \mathcal{C}$. The same thing can be said for $\mathcal{C}_{c_j}^-$ with respect to o_j. If relations that connect a concept to other concepts in $\mathcal{C}^-$ are ranked, equation (4.1) then becomes weighted such that $\left| \mathcal{C}_{c_i}^- \cap \mathcal{C}_{c_j}^- \right| = \sum_k \psi_{r_k}^{\mathcal{C}_{c_i}^- \cap \mathcal{C}_{c_j}^-}$ which is the sum of the weights of the relations that connect each concept in the intersection set with its connected concept c_i or c_j. For example, if

$\mathcal{C}^-_{c_i} = \{c_2, c_7, c_{12}, c_8\}$, and $\mathcal{C}^-_{c_j} = \{c_{13}, c_{23}, c_7, c_8\}$, then $\mathcal{C}^-_{c_i} \cap \mathcal{C}^-_{c_j} = \{c_7, c_8\}$ and $\left| \mathcal{C}^-_{c_i} \cap \mathcal{C}^-_{c_j} \right| = \psi_{r_{i7}} + \psi_{r_{i8}} + \psi_{r_{j7}} + \psi_{r_{j8}}$, such that $\psi_{r_{ik}}$ is the weight of the relation r_{ik} that connects c_i with c_k. The same process is done for the denominator in Eq. (4.1), such that $\left| \mathcal{C}^-_{c_i} \cup \mathcal{C}^-_{c_j} \right| = \sum_k \psi_{r_k}^{\mathcal{C}^-_{c_i} \cup \mathcal{C}^-_{c_j}}$. In subsequent chapters as we discuss ontology-based web recommendation, we compare the results of using this proposed measure for ontology-based recommendation with the case when only the number of separating is-a edges is used instead.

Example 4.3 (semantic distance): Consider the two concepts *Still Camera* and *Video Camera* from eMart's ontology in Figure 2.5. Assume that two pageviews are mapped to the two respective concepts c_7="*Still Camera*" and c_9="*Video Camera*", generating semantic objects o_7 and o_9, then the semantic distance γ_{o_7,o_9} is computed as follows:

$\gamma_{o_7,o_9} = 1 / \frac{\left| \mathcal{C}^-_{c_7} \cap \mathcal{C}^-_{c_9} \right|}{\left| \mathcal{C}^-_{c_7} \cup \mathcal{C}^-_{c_9} \right|}$,
$\mathcal{C}^-_{c_7} = \{Camera, Digital, Film\}$ (from Fig. 2.5) ,
$\mathcal{C}^-_{c_9} = \{Camera, Digital, Video\_Film\}$ (from Fig. 2.5) ,
$\mathcal{C}^-_{c_7} \cap \mathcal{C}^-_{c_9} = \{Camera, Digital\}$ and
$\left| \mathcal{C}^-_{c_7} \cap \mathcal{C}^-_{c_9} \right| = 2$,
$\mathcal{C}^-_{c_7} \cup \mathcal{C}^-_{c_9} = \{Camera, Digital, Film, Video\_Film\}$ and
$\left| \mathcal{C}^-_{c_7} \cup \mathcal{C}^-_{c_9} \right| = 4$,
$\Rightarrow \gamma_{o_7,o_9} = 1/(2/4) = 2$ $\qquad\qquad\qquad\qquad\qquad\qquad$ $\square$

Example 4.4 (semantic distance with weighted relations): Consider the same two concepts *Still Camera* and *Video Camera* from Example 4.3 above. Assume, again, that two pageviews are mapped to the two respective concepts c_7="*Still Camera*" and c_9="*Video Camera*", generating semantic objects o_7 and o_9. Assume that relations between concepts have weights as in Figure 4.3, then the semantic distance γ_{o_7,o_9} is computed as follows:

$\gamma_{o_7,o_9} = 1 / \frac{\left| \mathcal{C}^-_{c_7} \cap \mathcal{C}^-_{c_9} \right|}{\left| \mathcal{C}^-_{c_7} \cup \mathcal{C}^-_{c_9} \right|}$,
$\mathcal{C}^-_{c_7} = \{Camera, Digital, Film\}$ (from Fig. 4.3) ,
$\mathcal{C}^-_{c_9} = \{Camera, Digital, Video\_Film\}$ (from Fig. 4.3) ,
$\mathcal{C}^-_{c_7} \cap \mathcal{C}^-_{c_9} = \{Camera, Digital\}$ and
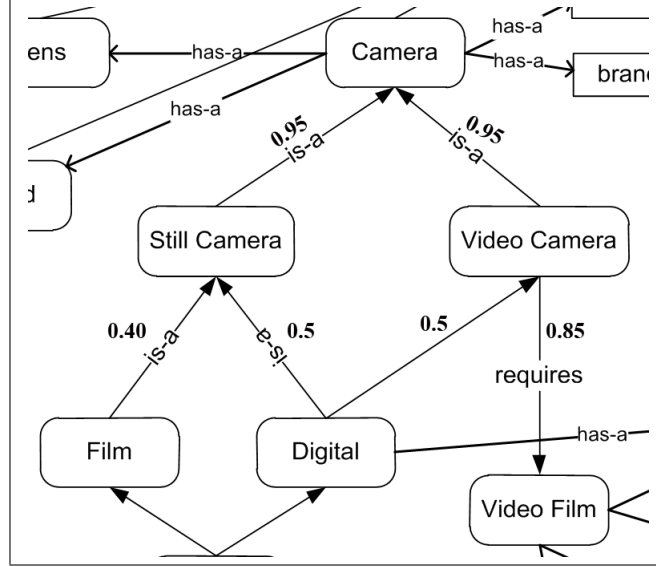
Figure 4.3: Example of weighted relations in the ontology, weights are boldface fractions. This is an enlarged part of the ontology in Figure 2.5 with assumed weights labeled on relation edges.

$\left| \mathfrak{C}_{c_7}^{-} \cap \mathfrak{C}_{c_9}^{-} \right| = \sum_k \psi_{r_k}^{\mathfrak{C}_{c_7}^{-} \cap \mathfrak{C}_{c_9}^{-}}$

$= \psi_{r_{71}} + \psi_{r_{91}} + \psi_{r_{72}} + \psi_{r_{92}}$ (where c_1="Camera" and c_2="Digital")

$= 0.95 + 0.5 + 0.95 + 0.5 = 2.9$,

$\mathfrak{C}_{c_7}^{-} \cup \mathfrak{C}_{c_9}^{-} = \{Camera, Digital, Film, Video\_Film\}$ and

$\left| \mathfrak{C}_{c_7}^{-} \cup \mathfrak{C}_{c_9}^{-} \right| = \sum_k \psi_{r_k}^{\mathfrak{C}_{c_7}^{-} \cup \mathfrak{C}_{c_9}^{-}},$

$= \psi_{r_{71}} + \psi_{r_{91}} + \psi_{r_{72}} + \psi_{r_{92}} + \psi_{r_{73}} + \psi_{r_{94}}$ (where c_3="Film" and c_4="Video_Film")

$= 0.95 + 0.5 + 0.95 + 0.5 + 0.4 + 0.85 = 4.15$

$\Rightarrow \gamma_{o_7,o_9} = 1/(2.9/4.15) = 1.43$ $\qquad\qquad$ □

A disadvantage of intensional approaches is their reliance on good modelling habits of people constructing the ontologies, as they require involvement from the ontology engineer, e.g., to put weights on edges. On the other hand, they have an advantage over extensional methods that require the presence of an unbiased population of instance data which is not always available, and is unfit for web recommendation applications in which items are not semantically pre-annotated [Hu et al., 2007]. For a survey and comparison between different semantic similarity and distance measures, the reader can refer to [Hu et al., 2007] and Cha [2007].

Algorithm 2 *getObject()* procedure for mapping pageviews from the web log to semantic object instances of ontology concepts

Input: Web page p_i,
   [optional] Domain Ontology $\mathcal{O}$,
Output: Set *so* of semantic object(s) represented by web page p_i ,
Algorithm:

 1: Initialize $so = \{\}$
 2: **if** an ontology $\mathcal{O}$ is provided **then**
 3:    /*mapping a semantic object to an ontology concept*/
 4:    Retrieve ontology class information c from annotation in given page p_i
 5:    create a semantic object o_i as a tuple $o_i =< pg_i, c >$, where pg_i is the URL of page p_i, and c is the class to which it is mapped
 6:    add semantic object o_i to the set *so* of semantic objects represented by page p_i
 7: **else**
 8:    build ontology $\mathcal{O}$, by calling function $OntoCraft()$ [see algorithm 1]
 9:    Search for $< meta\ name = "keywords"\ content = "keyword_1, keyword_2, \ldots" >$ tag in HTML code of web page p_i
10:    retrieve keyword(s) from the tag and add to list K
11:    /*mapping*/
12:    **if** there is only one keyword in K **then**
13:       /*one-to-one*/
14:       map this keyword to the ontology, by executing lines 5-6 above
15:    **else**
16:       {
17:       /*many-to-one*/
18:       for all keywords in K, find common ancestor by climbing up the hierarchy in ontology $\mathcal{O}$
19:       this common ancestor is now c, the class to be mapped to
20:       do the mapping as in lines 5-6 above
21:       }
22:       { /* or many-to-many*/
23:       **for** each keyword $keyword_k$ in K list of keywords **do**
24:          Map $keyword_k$ to its corresponding ontology class c_j in the ontology
25:          create a semantic object o_{i_k} as a tuple $o_{i_k} =< pg_i, c_k >$, where pg_i is the URL of page p_i, and c_k is the class to which $keyword_k$ is mapped
26:       **end for**
27:       add the tuple $< o_{i_1}, o_{i_2}, \ldots, o_{i_k} >$, resulting from previous for loop, to the set *so* of semantic objects
28:       }
29:    **end if**
30: **end if**
31: **return** set of semantic objects *so*
    end

Chapter 5

Semantics-aware Sequential Pattern Mining

SemAware integrates semantic information into Sequential Pattern Mining (SPM), this information that has been transformed into semantic-rich user transactions in the preprocessing phase is now used in the core of the adopted SPM algorithm, for effective pruning of the search space, and to enhance the scalability and performance of these algorithms. This work is published in Mabroukeh and Ezeife [2009b].

5.1 The Maximum Semantic Distance

The semantic distance matrix $\mathcal{M}$ is used during the pruning process to reduce the search space and minimize the number of candidate frequent sequences, minimizing as well the number of database scans and support counting processes. As presented before, preprocessing extracts all the semantic objects represented by pageviews in web log $\mathcal{W}$. Consider the running example in Table 4.1, page p_2 can be mapped to object o_2, representing an instance of the class of *Digital* Still Camera in the ontology of Figure 2.5, the "Canon PowerShot A2000 IS". Table 4.3 shows a mapping between the web pages from the sequence database of Table 4.1 being mined and the ontology of Figure 2.5, as a result of the preprocessing described. During mapping of web pages to their corresponding concepts, the semantic distance is computed and stored in the semantic distance matrix $\mathcal{M}$, e.g., Figure 5.1. As stated before and in order to validate the thesis hypothesis, we want to investigate the effect of domain semantics on SPM. Let us use this semantic information to prune the search space of these algorithms in a similar way as minimum support threshold is used. For this, we introduce the *maximum semantic distance*.

$$\mathcal{M} = \begin{bmatrix} & o_1 & o_2 & o_3 & o_4 & o_5 \\ o_1 & 0 & 1 & 5 & 1 & 3 \\ o_2 & 1 & 0 & 2 & 2 & 3 \\ o_3 & 3 & 1 & 0 & 4 & 8 \\ o_4 & 2 & 5 & 4 & 0 & 8 \\ o_5 & 10 & 9 & 10 & 12 & 0 \end{bmatrix}$$

Figure 5.1: Example Semantic distance matrix used in running example from Table 4.2.

Definition 9 (Maximum Semantic Distance η). *The maximum semantic distance η is a constraining value which represents the maximum allowed semantic distance between any two semantic objects.* ☐

Maximum semantic distance is inversely proportional to the maximum level of relatedness a user would allow between any two semantic objects. It can be user-specified (i.e., a user with enough knowledge of the used ontology can specify this value) or it can be automatically calculated from the minimum support value specified for the mining algorithm, by applying it as a restriction on the number of edges in the ontology graph. For example, if the minimum support used in the mining algorithm is 5% and the number of edges in the ontology is 60 edges, then $\eta = 3$, meaning that the maximum semantic distance allowed between any two classes in the ontology is only 3, $\eta = min\_sup \times |\mathcal{R}|$. The idea behind this calculation is that since η is a maximum measure of distance in the ontology that is directly related to the number of edges (i.e., relations), then it is logical to use min_sup as a restriction on this distance. In the basic form this distance is a simple count of edges in the ontology. So if η is the mean of semantic distances then this means that the maximum allowed semantic distance is half way through the ontology. In the experiments to follow, we show that the optimal value for η is the mean of the semantic distances in the matrix $\mathcal{M}$. This is an empirical study that supports the assumption, and is not surprising given the direct relation between semantic distance and the number of edges in the ontology. More future work can be conducted to investigate the relation and mining parameters that lead to an optimal value of η.

5.2 Procedure and Example of Semantic Pruning of the Search Space

Sequential Pattern Mining algorithms, especially Apriori-based algorithms like AprioriAll [Agrawal and Srikant, 1995] and GSP [Srikant and Agrawal, 1996] employ the generate-and-

test procedure, in which two (*k-1*)-sequences are joined to produce a candidate *k*-sequence at the *k*th iteration of the algorithm. Then the list of all candidate *k*-sequences is tested against the minimum allowed support *min_sup*, where all candidates with support less than *min_sup* are pruned from the search space. Algorithm 3 shows a general procedure for a semantics-based SPM algorithm with a replacement generate-and-test procedure that uses η to prune candidate sequences, such that if the semantic distance between the two (*k-1*)-sequences is more than an allowed maximum semantic distance η, then the candidate *k*-sequence is pruned from the search space without the need for support counting. Support counting requires a full scan of the sequence database for each candidate sequence. Function $SemJoin()$ implementation in Algorithm 3 is specific to the join procedure of the sequential pattern mining algorithm adopted in SemAware for a specific application, an example is SemApJoin() in Figure 5.2, which is used in *AprioriAll-sem* (a proposd semantics-aware variation of AprioriAll).

Algorithm 3 General algorithm for Semantics-aware Apriori-based Sequential Pattern Mining.

SemAwareSPM(*M*, *S*, η, *min_sup*)

Input: sequence database *S*,
  semantic distance matrix *M*,
  maximum semantic distance η
  minimum support *min_sup*

Output: Semantic-rich frequent sequences

Algorithm:

1: Scan database *S* to find the set of frequent 1-sequences, $L_1 = \{s_1, s_2, \ldots, s_n\}$.
2: *k*=1,
3: $C_1 = L_1$
  {Apply any apriori-based sequential pattern mining algorithm using η to prune the search space, as follows.}
4: **repeat**
5:   *k*++
6:   **for** $L_{k-1} \bowtie L_{k-1}$ **do**
7:     $\forall\ s_i,\ s_j$ such that $s_i,\ s_j \in L_{k-1}$
8:     $C_k \leftarrow C_k \bigcup SemJoin(s_i, s_j)$
9:   **end for**
10:   $L_k = \{c \in C_k\ :\ support(c) \geq min\_sup\}$
11: **until** $L_{k-1} = \phi$
12: **return** $\bigcup_k L_k$

Example 5.1 (AprioriAll-sem): Consider the semantic-rich sequence database in Table 4.2, transactions similar to t_1 are referred to as semantic-rich user transactions. The

semantic distance between any two objects can be computed faster than support counting that requires multiple scans of the large sequence database in some cases (e.g., Apriori-based algorithms) or more memory space in others (e.g., Tree Projection algorithms). Finding semantic similarity between any two objects is a matter of looking up the semantic distance matrix $\mathcal{M}$. In this example, we first apply an Apriori-based sequential pattern mining algorithm on the given sequence database, AprioriAll is used with *absolute minimum support* of 3 (*min_sup*=3). It scans the database several times to find frequent itemsets of size k at each kth iteration (starting from k=2). It also has the generate-and-test feature (see survey in Section 2.3.2) by performing the Apriori-generate join [Agrawal and Srikant, 1995] procedure to join L_{k-1} with itself to generate C_k, the set of candidate sequences in the kth-iteration, then prunes sequences in C_k which have subsequences not in L_{k-1}, creates L_k by adding all sequences from C_k with support $\geq$ *min_sup* until there are no more candidate sequences. The proposed *SemApJoin* procedure uses semantic distance for pruning candidate sequences, such that an object o_i is not affixed to the sequence s if $\gamma_{o_i o_j} > \eta$, where o_j is the last object in s (assume in this example that η=3). Figure 5.2 shows the details of *SemApJoin* which replaces Apriori-generate function in *AprioriAll-sem*.

---

insert into C_k
select p.litemset$_1$, ..., p.litemset$_{k\text{-}1}$, q.litemset$_{k\text{-}1}$
from L_{k-1} p, L_{k-1} q
where (p.litemset$_1$=q.litemset$_1$, ..., p.litemset$_{k\text{-}2}$=q.litemset$_{k\text{-}2}$)
   AND
      $\gamma_{p.litemset_{k-1}, q.litemset_{k-1}} \leq \eta$

Figure 5.2: *SemApJoin* procedure with semantic pruning of candidate sequences.

Following with AprioriAll, the set of candidate sequences C_1 for the database in Table 4.2 is C_1= {o_1:5, o_2:5, o_3:2, o_4:2, o_5:4}, where each item is represented with its support in the form *sequence:support count*. After pruning, the list of frequent 1-sequences is L_1={o_1:5, o_2:5, o_5:4}. The original Apriori-generate function now generates candidate sequences at k=2, C_2=$L_1 \bowtie_{ap-gen} L_1$= {o_1o_1, o_1o_2, o_1o_5, o_2o_1, o_2o_2, o_2o_5, o_5o_1, o_5o_2, o_5o_5}, on the other hand, *SemApJoin* function generates a smaller list of $C_2 = L_1 \bowtie_{SemApJoin} L_1$ = {o_1o_1, o_1o_2, o_1o_5, o_2o_1, o_2o_2, o_2o_5, o_5o_5} in which o_5o_1 and o_5o_2 are not generated simply because the semantic distance between o_5 and both o_1 and o_2 is larger than the maximum distance allowed η. This is found by consulting the semantic distance matrix in Figure 5.1. Next,

AprioriAll computes support count for each candidate sequence, resulting in $C_2 = \{o_1 o_1:2,$ $o_1 o_2:4,\ o_1 o_5:4,\ o_2 o_1:3,\ o_2 o_2:3,\ o_2 o_5:4,\ o_5 o_5:0\}$, the algorithm then prunes all sequences in C_2 which do not have frequent subsequences (i.e. their subsequences do not appear in L_1), after pruning, the list of frequent 2-sequences L_2 becomes $L_2 = \{o_1 o_2:4,\ o_1 o_5:4,\ o_2 o_1:3,$ $o_2 o_2:3,\ o_2 o_5:4\}$. The original Apriori-generate function now generates candidate sequences at $k=3$, $C_3 = L_2 \bowtie_{ap-gen} L_2 = \{o_1 o_2 o_2,\ o_1 o_2 o_5,\ o_1 o_5 o_2,\ o_1 o_5 o_5,\ o_2 o_1 o_1,\ o_2 o_1 o_2,\ o_2 o_1 o_5,\ o_2 o_2 o_1,$ $o_2 o_2 o_2,\ o_2 o_2 o_5,\ o_2 o_5 o_1,\ o_2 o_5 o_2,\ o_2 o_5 o_5\}$, on the other hand, $SemApJoin$ will generate a smaller list of $C_3 = L_2 \bowtie_{SemApJoin} L_2 = \{o_1 o_2 o_2,\ o_1 o_2 o_5,\ o_1 o_5 o_5,\ o_2 o_1 o_1,\ o_2 o_1 o_2,\ o_2 o_1 o_5,\ o_2 o_2 o_1,$ $o_2 o_2 o_2,\ o_2 o_2 o_5,\ o_2 o_5 o_5\}$ in which $o_1 o_5 o_2$, $o_2 o_5 o_1$ and $o_2 o_5 o_2$ are not generated, because the semantic distance between o_5 and both o_1 and o_2 is larger than η. Next, AprioriAll computes support count for each candidate sequence resulting in $C_3 = \{o_1 o_2 o_2:1,\ o_1 o_2 o_5:3,$ $o_1 o_5 o_5:0,\ o_2 o_1 o_1:1,\ o_2 o_1 o_2:1,\ o_2 o_1 o_5:2,\ o_2 o_2 o_1:2,\ o_2 o_2 o_2:0,\ o_2 o_2 o_5:2,\ o_2 o_5 o_5:0\}$. After pruning L_3 becomes $L_3 = \{o_1 o_2 o_5\}$ and the algorithm stops, resulting in the set of all frequent semantic objects $fo = \bigcup_k L_k = \{o_1,\ o_2,\ o_5,\ o_1 o_2,\ o_1 o_5,\ o_2 o_1,\ o_2 o_2,\ o_2 o_5,\ o_1 o_2 o_5\}$. One can notice that $SemApJoin$ allowed the mining algorithm to reduce the search space, up to 23% in this example.

Example 5.2 (GSP-sem): This example mines the sequence database of Table 4.2 using GSP [Srikant and Agrawal, 1996] and the semantics-aware GSP-sem. By applying GSP, The first pass on the database determines the support of each item to find frequent 1-sequences based on $min\_sup=3$. The first scan over the table at $k=1$ generates the set of candidate 1-sequences $C_1 = \{o_1:5,\ o_2:5,\ o_3:2,\ o_4:2,\ o_5:4\}$, giving the set of frequent 1-sequences $L_1 = \{o_1,\ o_2,\ o_5\}$. These sequences provide a *seed set* L_k which is used to generate next level candidate sequences in the next pass at $k+1$. The next C_{k+1} candidate set is generated by performing a GSP-join of L_k on itself. The GSP-join, like the Apriori-generate join requires that two sequences in L_k join with each other. A sequence s_1 (e.g., *ab*) in L_k joins with another sequence s_2 (e.g., *ba*) in L_k if the subsequence obtained by dropping the first (*k-1*) items of s_1 is the same as the subsequence obtained by dropping the last (*k-1*) items of s_2, the candidate sequence generated is the sequence s_1 extended with the last item in s_2 (e.g., *aba* is the result of joining *ab* and *ba*) and is added to C_k. According to this, the candidate set C_2 of 2-sequences is $C_2 = \{o_1 o_1,\ o_1 o_2,\ o_1 o_5,\ o_2 o_1,\ o_2 o_2,\ o_2 o_5,\ o_5 o_1,$ $o_5 o_2,\ o_5 o_5,\ (o_1 o_2),\ (o_1 o_5),\ (o_2 o_5)\}$, where parenthesis denote contiguous sequences (i.e., no time gaps). Applying semantics-aware GSP-sem, we get $C_2 = \{o_1 o_1,\ o_1 o_2,\ o_1 o_5,\ o_2 o_1,\ o_2 o_2,$ $o_2 o_5,\ o_5 o_5,\ (o_1 o_2),\ (o_1 o_5)\}$, in which $o_5 o_1$ and $o_5 o_2$ are not generated; because the semantic distance between o_5 and both o_1 and o_2 is larger than the maximum distance allowed η. The algorithm then scans the sequence database for each candidate sequence to count its support, after the pruning phase, to get $L_2 = \{o_1 o_2:3,\ o_1 o_5:4,\ o_2 o_1:3,\ o_2 o_2:3,\ o_2 o_5:4,\ (o_1 o_2):3\}$,

now GSP-join L_2 with L_2 to get C_3={$o_1o_2o_1$, $o_1o_2o_2$, $o_1o_2o_5$, $o_2o_1o_2$, $o_2o_1o_5$, $o_2(o_1o_2)$, $o_2o_2o_1$, $o_2o_2o_5$, $o_2o_2o_2$, $(o_1o_2)o_1$, $(o_1o_2)o_2$, $(o_1o_2)o_5$}, and so on until C_k={} or L_{k-1}={}. The set of mined frequent sequences is eventually $fs = \bigcup_k L_k$. To show an example of pruning the contiguous subsequence in GSP, suppose we are given L_3={$(o_1o_2)o_3$, $(o_1o_2)o_4$, $o_1(o_3o_4)$, $(o_1o_3)o_5$, $o_2(o_3o_4)$, $o_2o_3o_5$}, then C_4={$(o_1o_2)(o_3o_4)$, $(o_1o_2)o_3o_5$}, and $(o_1o_2)o_3o_5$ is pruned because its contiguous subsequence $o_1o_3o_5$ is not in L_3. While applying semantics-aware GSP-sem to generate C_4 results in an empty set, since $(o_1o_2)(o_3o_4)$ and $(o_1o_2)o_3o_5$ are pruned; because the semantic distance between o_3 and both o_4 and o_5 is larger than η.

5.3 Computational Complexity

We discussed in Section 2.3.2 that the space complexity of Aproiri-based algorthims is factorial in n, where n is the number of frequent 1-sequences. However, in GSP-sem and AprioriAll-sem this complexity is reduced by the semantic joining function (e.g., SemApJoin in AprioriAll-sem), which on its turn depends on the value of η. In Apriori-based algorithms at every kth iteration, an approximate number of $\binom{n}{k}$ candidate sequences is generated, call it c. Assuming that l number of candidate sequence was not generated due to pruning by the semantic joining function, then the number of generated candidate sequences at each kth iteration after semantic pruning is $p = c - l$, that is $p < c$. The best case is achieved when $\eta = 1$, and the worst case is when η is too large leading to $p = c$. One can say that the number of candidate sequences generated in AprioriAll-sem and GSP-sem is approximately $\sum_k \binom{n}{k} - l_k$, where k is bound by the size of the largest frequent sequence generated, and l_k is the number of candidate k-sequences that were not generated due to semantic pruning.

5.4 Experimental Evaluation

5.4.1 Methodology

We tested two Apriori-based semantics-aware sequential pattern mining algorithms (*GSP-sem* and *AprioriAll-sem*), and one Pattern-Growth algorithm (*PLWAP-sem*) on a 2.66GHz Intel Core2 Quad computer with 4 gigabytes memory running Windows Vista 32-bit, using two classes of datasets, synthetic and staged. The synthetic data sets are generated using the IBM resource data generator code [Agrawal et al., 1993]. The staged dataset is manually constructed to resemble a real web log. The following parameters are used to generate the data sets as described in Agrawal and Srikant [1995]: |D| represents the

number of sequences in the database, |C| is the average length of the sequences, |S| is the average length of maximal potentially frequent sequence, |N| is the number of events, |T| is the average number of items per transaction. A medium sized data set described as C10T6N40S4D50K and a large sized data set described as C15T8N100S4D100K. These are mined at different minimum support values, low minimum support of 1% and regular minimum support of 10%, while the maximum semantic distance is fixed at η=10. Semantic distances are entered into the semantic distance matrix $\mathcal{M}$ as random numbers drawn from a normal distribution with mean=50 and variance=40. CPU execution time is reported by the program of each algorithm in minutes, while physical memory usage is measured in Kilobytes (KB). All algorithms are implemented in C++. The programs are implemented in a way such that they report their own CPU and Memory usage, which is later verified using the Microsoft .NET CLR Profiler. *PLWAP-sem* was implemented as a variation of *PLWAP* [Ezeife and Lu, 2005], such that semantic distance is checked against each pair of items when the tree is built, and an object o_i is not added to its location in the tree if $\gamma_{o_i o_j} > \eta$, where o_j is the object that comes before o_i in the sequence being added to the tree.

5.4.2   Results

Tables 5.1 and 5.2 show the performance of the proposed algorithms. It is found that semantic-aware algorithms, namely, *GSP-sem* and *AprioriAll-sem*, require on the average only 26% of the search space, although the semantic distance matrix is stored in the form of a direct access 2-dimensional array. The main reason for keeping an array data structure is the ease and speed of accessing the indexed array to check for semantic distance between any two objects. When the mining process starts finding size-2 frequent sequences and larger, non-semantic algorithms' memory consumption starts picking up more than that of semantic-aware algorithms. A good increase in mining speed was also noticed. *GSP-sem* and *AprioriAll-sem* are 3-4 times faster than the regular GSP and AprioriAll algorithms, while PLWAP-sem is 15 times faster than PLWAP

To test the scalability of the semantic algorithms against different values for η, a sparse synthetic data set is used, C8T5S4N100D200K. Sparse data sets are characterized by having a large number of unique items |N| (e.g., a large number of web pages in a given web site) and shorter customer sequences |C| and |T| (e.g., short user browsing sessions), such data sets represent typical characteristics of web logs. The results show enhanced performance at smaller values for η, as expected, the result of pruning more candidate sequences during mining. To find the optimal value for η, that will produce mining results similar to non-

Table 5.1: Algorithms execution time on synthetic data sets.

Execution time in minutes		
	Data sets	
Algorithm	|D|=50K	|D|=100K
GSP-sem	39	169
GSP	196	>12 Hrs.
AprioriAll-sem	107	314
AprioriAll	314	>12 Hrs.
PLWAP-sem	0.2	1
PLWAP	1	20

Table 5.2: Algorithms memory usage on synthetic data sets.

Memory in KB		
	Data sets	
Algorithm	|D|=50K	|D|=100K
GSP-sem	604	732
GSP	1,780	4,800
AprioriAll-sem	860	936
AprioriAll	2,300	2,560
PLWAP-sem	151	183
PLWAP	356	426

semantic-aware algorithms, a real web log was constructed to resemble a web log of *eMart*, with 50,000 transactions and 100 unique web pages. The semantic distance matrix was produced manually, from a given ontology, and fed to *GSP-sem* to mine the data set. Table 5.3 shows execution time results and memory usage for different values of η between 5 and 30. In all cases, minimum support was fixed at 1%. GSP algorithm consumes 2,364Kbytes of memory for this sparse data set at this given minimum support and requires more than 8 hours of mining time. The table shows enhanced performance as the value of η becomes smaller, which results in pruning more candidate sequences during mining, thus speeding up the process and using less memory. It is found that values for η between 3 and 4 allow *GSP-sem* to produce same frequent sequences as GSP, and yet still use 38% less memory, and run 2.8 times faster than GSP, while for PLWAP-sem, values of η between 7 and 9 allow the algorithm to produce same frequent patterns with 57% less memory.

To look into the effectiveness of the proposed semantics-aware algorithms, a real web log

Table 5.3: Semantics-aware GSP-sem and AprioriAll-sem scalability while changing η. Minimum support is set at 1%.

Max. Sem. Dist. η	GSP-sem		AprioriAll-sem		PLWAP-sem	
	Memory (KB)	CPU time (min.)	Memory (KB)	CPU time (min.)	Memory (KB)	CPU time (min.)
5	712	43	636	43	103	1.5
10	912	87	776	130	408	15
20	1,336	196	1,260	174	438	20
30	1,712	273	1,636	259	445	21

with an expert domain ontology is used to conduct experiments and find the percentage of resulting frequent patterns which match the patterns of the case of non-semantics-aware mining. In this case, we were able to construct a sequence database from the publicly available MovieLens dataset[1]. The semantic distance matrix is constructed from the publicly available movie ontology MO-Movie Ontology [Bouza, 2010]. The dataset was mined by the semantics-aware PLWAP-sem algorithm. Figure 5.3 shows the percentages of frequent patterns that match the case of non-semantics-aware PLWAP mining of the same dataset, in two different cases. The first case (dotted line in the figure, refer to it as Case-1) represents results when using PLWAP-sem with a shallow ontology. That is, the semantic distance is computed using only is-a relations by counting the number of separating edges between ontology concepts. The second case (the solid line in the figure, refer to it as Case-2) shows the percentages when the proposed semantic distance measure in Equation (4.1) is used, this is the measure that incorporates all semantic relations and relation weights. This should support the thesis hypothesis in Section 3.2 that the use of domain ontology adds to the effectiveness of the algorithms and enhances scalability, in that, Figure 5.3 shows that in Case-2 the resulting frequent patterns match those of the real user behavior (found by using PLWAP without semantics) to a large extent (90%-100%) for early values of the allowed maximum semantic distance, i.e. $\eta \geq 7$. Thus, allowing for more pruning to take place while maintaining a good match. On the other hand, pruning does take place in Case-2, but matches start to appear later, for values of $\eta \geq 10$. This means that the proposed semantic distance measure in Equation (4.1) provides for effective pruning that is sensitive to actual user behavior, i.e., more matching patterns while having low values for η.

---

[1]available at http://www.grouplens.org/node/73
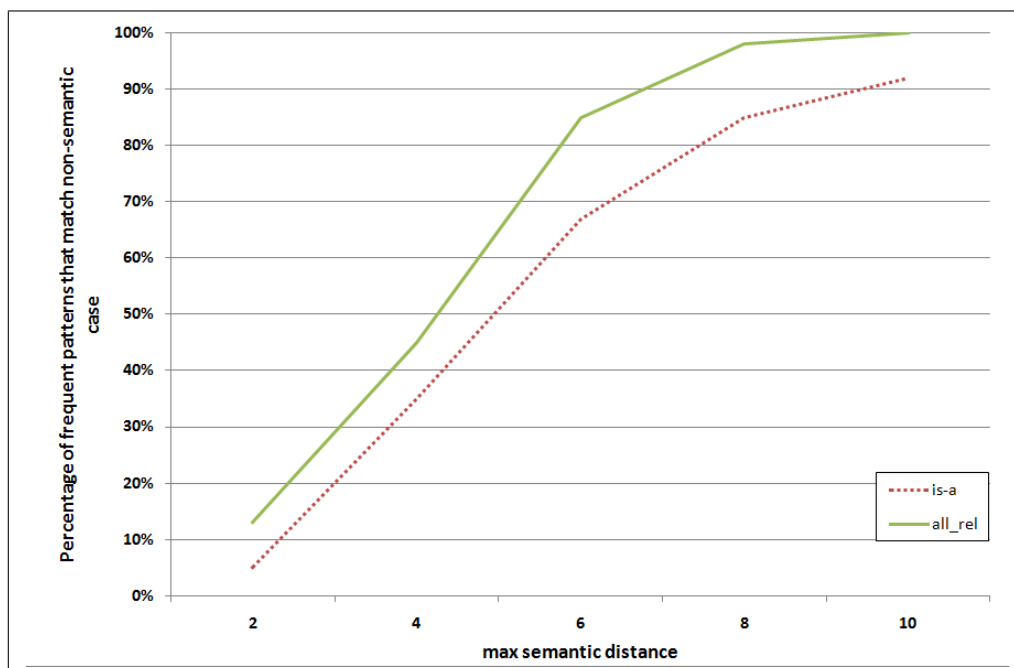
Figure 5.3: Comparison between mining using a shallow ontology (is-a) and using a domain ontology with all semantic relations (all_rel) on the MovieLens dataset.

5.5 Summary

A method is proposed to enhance SPM algorithms by using semantic distance measures, to prune candidate sequences and partially relief the algorithm from support counting in order to solve the problem of scalability in SPM, and a general semantics-aware mining procedure is presented. Two Apriori-based semantics-aware SPM algorithms are implemented, namely AprioriAll-sem and GSP-sem, and PLWAP-sem as a semantics-aware variation of the Pattern-Growth/Early-Pruning hybrid algorithm PLWAP [Ezeife and Lu, 2005]. Experimental results show that the proposed algorithms can perform faster than the regular algorithms with less memory requirement, when a low min_sup and low η values are used, without compromising the quality of frequent patterns generated. It is also shown that using domain ontology to measure the semantic distance as proposed by this thesis (eq. (4.1)) for these algorithms is more effective that using a shallow ontology, since the earlier is more sensitive to actual user behavior and the semantic relations among items. Future work needs to look at the effect of many-to-one and many-to-many mappings of semantic objects on the mining process and the efficiency of semantic pruning in these two different cases.

Chapter 6

Semantics-aware Next Page Request Prediction

Predicting user's next page request on the World Wide Web is a problem that affects web server's cache performance and latency. Different methods exist that can look at the user's sequence of page views, and predict what next page the user is likely to view so it can be prefetched. The common method for predicting next page request, is to model the user's accessed web pages as a Markov process with states representing the accessed web pages and edges representing transition probabilities between states computed from the given user sequence in the web log. The work proposed here shows yet another validation that semantic information can be integrated into the mining core of a predictive model for recommendation results and to solve the *problems of contradicting predictions* and *tradeoff* in Markov models. This work is published in Mabroukeh and Ezeife [2009a] and partially in Mabroukeh and Ezeife [2009b].

6.1 Markov Models for Prediction

Given a sequence of web pageviews generated by a user browsing the world wide web. $\mathcal{W} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_i, \ldots, \mathcal{P}_l\}$, where $\mathcal{P}_i$ is a random variable representing the ith pageview in $\mathcal{W}$. The actual web page in a user's web session will be represented by p_i. The problem of next page request prediction is to predict the web page that will be accessed next, i.e. $\mathcal{P}_{l+1}$.

To model the transition between different web pages in a Markov process, the probability that a user accesses a certain web page next (or a certain item) is based on the current state that he is visiting, resulting in a 1st-order Markov model, or the previous k states in

the sequence, resulting in a kth-order Markov model. The probability of moving to state S_2 in a Markov model given the current state S_1 is the conditional probability $P(S_2|S_1)$. For example, the probability of the sequence $< p_2p_5 >$ happening, if $\mathcal{P}_i =< p_2 >$ and $\mathcal{P}_{i+1} =< p_5 >$, is the conditional probability of accessing $\mathcal{P}_{i+1}$ after $\mathcal{P}_i$ as follows:

$$Prob(\mathcal{P}_{i+1} = p_{i+1}|\mathcal{P}_i = p_i) = \frac{frequency(< p_ip_{i+1} >)}{frequency(< p_i >)} \tag{6.1}$$

Let S_j^k be a state containing k page views from $\mathcal{W}$, and l be the number of pages the user visited so far, $S_j^k = < p_{l-(k-1)}, p_{l-(k-2)}, \ldots, p_l>$. The probability of accessing a page p_i after the set of k pages S_j^k is estimated in a kth-order Markov model from a history web log (training data) as follows:

$$Prob(\mathcal{P}_i = p_i|S_j^k) = \frac{frequency(< S_j^k p_i >)}{frequency(< S_j^k >)} \tag{6.2}$$

Using $\mathcal{W}$, the page p_{l+1} that the user will access next is given by

$$p_{l+1} = \underset{p \in \mathbb{P}}{\arg\max}\{Prob(\mathcal{P}_{l+1} = p_{l+1}|\mathcal{P}_l, \mathcal{P}_{l-1}, \ldots, \mathcal{P}_{l-(k-1)})\} \tag{6.3}$$

where $\mathbb{P}$ is the set of all pages in the web site. The **argmax** operator returns the page with the highest conditional probability as a result of prediction. The *contradicting predictions* problem occurs when **argmax** returns more than one result with equal probabilities.

With S_j^k representing the states of the Markov model, a markov process is modeled as a directed acyclic graph in which every vertex represents a state corresponding to a pageview in the sequence, and edges labeled with probabilities representing transitions between the connected states according to equation (6.1), or (6.2) in the case of kth-order Markov model. All transition probabilities are stored in a transition probability matrix $\mathbf{P}_{n\times n}$, where n is the number of states in the model. This matrix contains the transition probabilities from every state to every other state in the Markov model.

During the prediction phase, and when the user is browsing a certain web page, say page a, the transition matrix $\mathbf{P}$ is consulted by the prediction system as a lookup table, in an implementation of equation (6.3), to see what next state has highest transition probability from the current state which represents a. The web page with highest probability of occurring next is output as the result of prediction.

6.2 Enriching the Transition Probability Matrix with Semantic Information

Semantic information in the form of semantic distance is also integrated in the prediction phase. Here, we introduce a novel way in which a 1st-order Markov model [Deshpande and Karypis, 2004] is enriched with semantic distance measures and used for predicting next page requests.

Semantic information can be used in a Markov model as a proposed solution to provide semantically interpretable and accurate prediction without using complicated kth-order or selective Markov models as discussed previously. The semantic distance matrix $\mathcal{M}$ is directly combined with the transition probability matrix $\mathbf{P}$ of a Markov model of the given sequence database, into a weight matrix $\mathbf{W}$. This weight matrix is consulted by the predictor software (like SemAware), instead of $\mathbf{P}$, to determine future page view transitions for caching or prefetching.

Definition 10 (Weight Matrix)**.** *The Weight Matrix* $\mathbf{W}$ *is the result of combining the semantic distance matrix* $\mathcal{M}$ *with the Markov transition probability matrix* $\mathbf{P}$.
$\mathbf{W}_{n \times n} : \mathcal{M} \oplus \mathbf{P} \longrightarrow w_{o_i,o_j}$, *where the combination function* $\oplus$ *is described as follows,*

$$
w_{o_i,o_j} = Prob(\mathcal{P}_{x+1}|\mathcal{P}_x) + \begin{cases} 1 - \dfrac{\frac{\gamma_{o_i,o_j}}{n}}{\sum\limits_{k=1}^{n} \gamma_{o_i,o_k}} & , \ \gamma_{o_i,o_j} > 0 \\[20pt] 0 & , \ \gamma_{o_i,o_j} = 0 \end{cases} \tag{6.4}
$$

, where $\mathcal{P}_{x+1} = p_{x+1}$ *and* $\mathcal{P}_x = p_x$ *and* o_i *is the semantic object from* p_x *and* o_j *is the semantic object from* p_{x+1}. $\qquad \qquad \square$

Definition 10 above defines a general combination function for the purpose of combing the probability matrix with the semantic distance matrix, and uses addition as an example of this combination. The goal is to convert the probability to a weight such that the shorter the semantic distance the higher this weight should be when combining the distance with probability, and the probability of moving from one state to another should be affected by the semantic distance between the two states and not other states, thus the direct addition of the distance to the probability. In detail, and in order to combine $\mathbf{P}$ and $\mathcal{M}$, $\mathcal{M}$ has to be normalized such that each entry is between 0 and 1. This is achieved by dividing each row entry by the row sum $\sum_{k=1}^{n} \gamma_{o_i,o_k}$. The next step would be to add both matrices together, in order to enrich $\mathbf{P}$ with semantic distance measures. But, the values in $\mathcal{M}$ represent

a distance, such that the higher the value, the more is the distance, which is inversely proportional with the required output weight (i.e., a greater distance should result in a smaller weight). To solve this problem the complement of the normalized $\mathcal{M}$ is what is actually added to **P**. The complement is found by subtracting each non-zero entry in the normalized $\mathcal{M}$ from 1. Finally, **W** is normalized by dividing the each row entry by the maximum value in that row, so the weight matrix can generically fit in any Markov-based prediction tool in place of **P**.

6.3 Procedure and Example of Semantic-Rich Markov Models

To create a 1st-order Markov model of the sequence database in Table 4.1 (the running example), we use equation 6.1, and consider the sequence as a training set to build the model and create the transition probability matrix. For ease of reference Table 4.1 is reproduced here as Table 6.1.

Table 6.1: Sequence Database from Table 4.1 for Markov model example.

Transaction ID	Sequence
t_1	$p_2 p_3 p_2 p_1 p_5$
t_2	$p_2 p_1 p_3 p_2 p_1 p_5$
t_3	$p_1 p_2 p_5$
t_4	$p_1 p_2 p_5 p_2 p_4$
t_5	$p_1 p_2 p_1 p_4$

Example 6.1 (Semantic-rich Markov models): The probability of going from p_2 to p_3 is 1/8 (or 0.125), and is calculated by dividing the frequency of $< p_2 p_3 >$ by the frequency of $< p_2 >$ from Table 6.1, while the probability of starting with state p_2 (i.e., going from S to state p_2) is 2/5 because 2 out of the 5 transactions in the database start with p_2. Figure 6.1 shows the complete Markov model after training, along with its transition probability matrix **P** in Figure 6.2. For an example prediction, assume that the user went through this sequence of page views $<p_2 p_5 p_1 p_3>$. Looking at **P** in Figure 6.2, there is a 100% probability that the user will next view page p_2. This shows an important shortcoming of using Markov models in next page prediction, as follows. If a new page has been introduced in the web site as a transitional page between page p_3 and page p_4, say page p_7. The system based on Markov model here cannot predict that the next page is p_7 instead of p_2 because p_7 was not introduced in training (cold start problem).

Figure 6.1: 1st-Order Markov model for Table 6.1.

$$
P=
\begin{array}{c|ccccc}
 & p_1 & p_2 & p_3 & p_4 & p_5 \\
\hline
p_1 & 0 & 0.43 & 0.14 & 0.14 & 0.28 \\
p_2 & 0.5 & 0 & 0.125 & 0.125 & 0.25 \\
p_3 & 0 & 1 & 0 & 0 & 0 \\
p_4 & 0 & 0 & 0 & 0 & 0 \\
p_5 & 0 & 0.25 & 0 & 0 & 0 \\
\end{array}
$$

Figure 6.2: Transition probability matrix for Markov model in Figure 6.1.

Another problem that could arise here is *contradicting prediction*, for illustration, assume in Figure 6.2 that $P(p_2|p_1) = P(p_5|p_1) = 0$, and notice that $P(p_3|p_1) = P(p_4|p_1)$, which means that there is an equal probability that a user views page p_3 or p_4 after viewing page p_1. Thus, the prediction capability of the system is not accurate in terms of which is more relevant to predict after p_1, and the prediction is ambiguous. Integration of the semantic distance matrix solves this problem as experiments prove in Section 6.6 . The transition matrix can be combined with the given semantic distance matrix $\mathcal{M}$ of Figure 6.3, resulting with $\mathbf{W}$ matrix in Figure 6.4 according to equation (6.4), as follows. First the semantic distance matrix is normalized by dividing each row by the sum of the values in that row, resulting in $\mathcal{M}'$. Then each item in the semantic distance matrix is subtracted from 1 (except zero values), resulting in $\mathcal{M}''$. In this way the complement of each semantic

distance value is calculated, this is the actual value that needs to be combined with the transition probability to show actual effect. Finally, $\mathbf{P}'$ and $\mathcal{M}''$ are combined by matrix addition according to equation (6.4), resulting in $\mathbf{W}$. The resulting combined matrix $\mathbf{W}$ is no longer a stochastic matrix because the matrix does not represent simple probability any more. But for this matrix to be used in place of a transition probability matrix, its entries must be normalized in the range 0 to 1, as described before, by dividing each row entry by the maximum entry in its row. The resulting matrix provides weights for moving from one

$$\mathcal{M} = \begin{bmatrix} & p_1 & p_2 & p_3 & p_4 & p_5 \\ p_1 & 0 & 1 & 5 & 1 & 3 \\ p_2 & 1 & 0 & 2 & 2 & 3 \\ p_3 & 5 & 2 & 0 & 4 & 8 \\ p_4 & 1 & 2 & 4 & 0 & 8 \\ p_5 & 3 & 3 & 8 & 8 & 0 \end{bmatrix}$$

Figure 6.3: Semantic distance matrix to be combined with Markov Transition Probability Matrix.

$$W = \begin{bmatrix} & p_1 & p_2 & p_3 & p_4 & p_5 \\ p_1 & 0 & 1.33 & 0.64 & 1.04 & 0.98 \\ p_2 & 1.37 & 0 & 0.87 & 0.87 & 0.87 \\ p_3 & 0.74 & 1.89 & 0 & 0.79 & 0.58 \\ p_4 & 0.93 & 0.87 & 0.73 & 0 & 0.47 \\ p_5 & 0.86 & 1.11 & 0.64 & 0.64 & 0 \end{bmatrix}$$

(a) Before normalization

$$W = \begin{bmatrix} & p_1 & p_2 & p_3 & p_4 & p_5 \\ p_1 & 0 & 1.00 & 0.48 & 0.78 & 0.74 \\ p_2 & 1.00 & 0 & 0.64 & 0.64 & 0.64 \\ p_3 & 0.39 & 1.00 & 0 & 0.42 & 0.31 \\ p_4 & 1.00 & 0.94 & 0.79 & 0 & 0.51 \\ p_5 & 0.77 & 1.00 & 0.58 & 0.58 & 0 \end{bmatrix}$$

(b) After normalization

Figure 6.4: Weight matrix W resulting from combining $\mathcal{M}$ (Figure 6.3) with Markov transition matrix $\mathbf{P}$ according to Eq. 6.4.

state to another, that can be used in place of transition probabilities. Notice from Figure 6.4 that the weight of moving from p_1 to p_3 is no longer the same as moving from p_1 to p_4 as the transition probability matrix suggests in Figure 6.2, thus providing a well-informed and preferable prediction, in the sense that pages which are more semantically related to the current page are being predicted as the new pageview, rather than pages which might have weak semantic relation to the current page.

6.4 State Pruning in Selective Markov Models Using Semantic Distance Measures

As discussed before, using Markov models for prediction suffers from a number of draw-backs. As the order of the Markov model increases, so does the number of states and the model complexity. On the other hand, reducing the number of states leads to inaccurate transition probability matrix and lower coverage, thus less predictive power, and less accuracy. To counter the reduction in coverage, and as a solution to this tradeoff problem, various Markov models of differing order can be trained and used to make predictions in the All-Kth-Order Markov model [Pitkow and Pirolli, 1999], such that if the kth-order Markov model cannot make the prediction then the (k-1)th-order Markov model is tried, and so on. The problem with using the All-Kth-Order Markov model is the large number of states contributing to the complexity of the model and the latency of prediction, making it inappropriate for online prediction. It is actually observed that a 1st-order Markov model provides 83% the predictive power of an All-Kth-order Markov model while consuming only 4.2% of the space [Pitkow and Pirolli, 1999]. A considerable number of the states in an All-Kth-order Markov model might never be used during prediction, besides the fact that many states can be shared between the different Markov models being built.

On the other hand, selective Markov models (SMM) [Deshpande and Karypis, 2004] that only store some of the states within the model have also been proposed as a solution to the mentioned tradeoff problem. They start off with an All-Kth-Order Markov model, then a post pruning approach is used to prune out states that are not expected to be accurate predictors. The result is a model that has the prediction power of All-Kth-Order models with less space complexity and more prediction accuracy. Deshpande and Karypis [2004] provide three different criteria which might be used separately to prune states in the model before prediction, that is, frequency, confidence, and error. But they did not study the effect and the relation of domain knowledge and semantics on selective Markov models, neither did they try to combine the three pruning criteria into one pruning method.

In this thesis, the maximum semantic distance measure η is used for state pruning in Selective Markov models (SMM). In this case, an All-Kth-Order Markov model is first built as in Pitkow and Pirolli [1999], next states that do not contribute to the model, i.e. which have zero frequency, are pruned. Then, any state S_j^k, having $\gamma_{p_{l-(k-1)}, p_{l-(k-2)}} > \eta$, where l is the number of pages the user visited so far and j is a simple enumeration of the states in the model, will be pruned from the model. In our examples and experimentation, we limit our models to 3rd-Order Markov models, similar to Deshpande and Karypis [2004].

Example 6.2 (Pruning SMM): Table 6.2 shows the All-Kth-Order Markov model

for the sequence database in Table 6.2(a). This model consists of 1st, 2nd and 3rd -order models. To create the selective Markov model, states with a right arrow "$\rightarrow$" are pruned as they do not contribute to the model (i.e. they have no next state), and states with a double right arrow "$\Rightarrow$" are pruned since the semantic distance between the pages is higher than the maximum allowed semantic distance (assumimg $\eta = 2$). For example, state $S_{18}^2 = <p_5, p_2>$ is pruned since $\gamma_{p_5, p_2} = 3$ is greater than η based on the semantic distance matrix of Figure 6.3. In this example, 14 states are pruned from just the 2nd-order part of the selective model, 5 of which are pruned based on semantic distance, in total resulting in 70% reduction in just the 2nd-order state space of the model, over the All-Kth-order model.

6.5 Computational Complexity

The complexity for building a 1st-order Markov model and calculating the transition probability matrix $\mathbf{P}$ can be computed at a maximum of $O(n^2)$, where n is number of states (i.e. unique items in the sequence database). For an All-Kth-Order Markov model, the computation complexity is $O(n^{k+1})$, k being the maximum order in the model. On the other hand, for building the proposed semantically-pruned SMM, the number of states generated will be $n^2 + (\sum_{j=2}^{k} n^{j+1} - n^{j-1} l_{j-1}) - l_k$, where l_j is the number of pruned states. The prediction process requires $O(n^k)$ worst case, i.e., when no pruning has taken place.

6.6 Experimental Evaluation

6.6.1 Methodology

Experiments are carried out on three kinds of data sets. Two data sets, *DS-1* and *DS-2* are generated using the IBM resource data generator [Agrawal et al., 1993]. *DS-1* is a small data set resembling a web log of 5000 user sessions, while *DS-2* is a large data set resembling a web log of 80,000 user sessions. A third data set *DS-3* is a staged data set manually constructed to resemble *eMart*'s web log with 200,000 user sessions. Characteristics of the three data sets are described in Table 6.3.

Using these data sets, 1st-Order, 2nd-Order, and All$K$th-Order Markov models, along with frequency-pruned SMM are built for testing and comparison (with $k = 3$). Semantic information in the form of a semantic distance matrix $\mathcal{M}$ for each data set, is generated randomly from a normal distribution. While for *DS-3*, the semantic distance matrix is manualy constructed based on the given *eMart*'s ontology. Using this matrix, semantic-

Table 6.2: (a) Sample web log with user transactions. (b) Resulting transition probability matrix for 1st-Order Markov model. (c) Resulting transition probability matrix for 2nd-Order Markov model.

TransID	Access Sequence
t_1	$p_2 p_3 p_2 p_1 p_5$
t_2	$p_2 p_1 p_3 p_2 p_1 p_5$
t_3	$p_1 p_2 p_5$
t_4	$p_1 p_2 p_5 p_2 p_4$
t_5	$p_1 p_2 p_1 p_4$
t_6	$p_3 p_1 p_4 p_2 p_1 p_5$
t_7	$p_4 p_1 p_2 p_5 p_2 p_3 p_2 p_1$
t_8	$p_1 p_4 p_2 p_3 p_1 p_2 p_5$

(a)

1st-Order State	p_1	p_2	p_3	p_4	p_5
$S_1^1 = <p_1>$	0	0.38	0.08	0.23	0.23
$S_2^1 = <p_2>$	0.43	0	0.21	0.07	0.29
$S_3^1 = <p_3>$	0.40	0.60	0	0	0
$S_4^1 = <p_4>$	0.20	0.40	0	0	0
$S_5^1 = <p_5>$	0	0.29	0	0	0

(b)

2nd-Order State	p_1	p_2	p_3	p_4	p_5
$S_1^2 = <p_1, p_2>$	0.20	0	0	0	0.80
$\Rightarrow S_2^2 = <p_1, p_3>$	0	1.00	0	0	0
$S_3^2 = <p_1, p_4>$	0	0.67	0	0	0
$\rightarrow S_4^2 = <p_1, p_5>$	0	0	0	0	0
$S_5^2 = <p_2, p_1>$	0	0	0.17	0.17	0.50
$S_6^2 = <p_2, p_3>$	0.33	0.67	0	0	0
$\rightarrow S_7^2 = <p_2, p_4>$	0	0	0	0	0
$\Rightarrow S_8^2 = <p_2, p_5>$	0	0.50	0	0	0
$\Rightarrow S_9^2 = <p_3, p_1>$	0	0.50	0	0.50	0
$S_{10}^2 = <p_3, p_2>$	1.00	0	0	0	0
$\rightarrow S_{11}^2 = <p_3, p_4>$	0	0	0	0	0
$\rightarrow S_{12}^2 = <p_3, p_5>$	0	0	0	0	0
$S_{13}^2 = <p_4, p_1>$	0	1.00	0	0	0
$\Rightarrow S_{14}^2 = <p_4, p_2>$	0.50	0	0.50	0	0
$\rightarrow S_{15}^2 = <p_4, p_3>$	0	0	0	0	0
$\rightarrow S_{16}^2 = <p_4, p_5>$	0	0	0	0	0
$\rightarrow S_{17}^2 = <p_5, p_1>$	0	0	0	0	0
$\Rightarrow S_{18}^2 = <p_5, p_2>$	0	0	0.50	0.50	0
$\rightarrow S_{19}^2 = <p_5, p_3>$	0	0	0	0	0
$\rightarrow S_{20}^2 = <p_5, p_4>$	0	0	0	0	0

(c)

rich 1st-order Markov models and semantic-pruned SMM are also constructed for testing. A frequency threshold [Deshpande and Karypis, 2004] of 0 is used in the Selective markov models (SMM), while varying values for η are used in the semantic-pruned SMM.

Testing is done in the following way. First, every data set is divided into a training set, which is the first 75% sessions in the data set, and a test set, the remaining 25%. Then,

Table 6.3: Data sets used for experimental analysis.

Data set	# of Transactions	# of Unique pages	Ave. Trans. Length
DS-1	5000	113	2.5
DS-2	80000	200	5
DS-3	200000	155	8

in the training part, the described Markov models are construced from the data sets and the model size for each one is noted. *Model size* is defined here as the number of states in each model. The testing part is made similar to the method described in Deshpande and Karypis [2004], that is, every model is given a trimmed session from the test set for prediction, in which the last page of the session is hidden. The prediction made by the model is then compared with the hidden page of the session to compute the accuracy of the model. *Accuracy* represents the predictive power of the model and is measured as the percentage of successful predictions made.

The performance in these experiments is measured by model size and accuracy, the two factors in the tradeoff problem described previously. The goal is to find the best model that can provide accurate predictions while maintaining a comparatively small model size. Sometimes a model might not be able to provide a prediction due to two reasons. First, a contradicting prediction problem might occur. Second, the hidden page might not have been present in the training set. If that happens, it will output the web page with the highest frequency as a default prediction, in the case of selective or stand-alone Markov models, or it will depend on semantic distance measures to make an informed prediction that is considered semantically correct, as is the case in semantic-rich models.

6.6.2 Results

Semantic-rich 1st-order Markov models are found to totally eliminate the contradicting prediction problem in all of the data sets used. For example, testing *DS-3* using the 1st-order Markov model resulted in a number of 258 contradicting predictions, while running the same test using the semantic-rich 1st-order Markov model, resulted in 0 contradicting predictions. This is confirmed when testing with 2nd-Order and 3rd-Order models. Albeit that, theoriticaly, semantic-rich Markov models do not totally eliminate the problem, as there is still the chance that two or more pages have the same probability of occuring and have the same semantic distance from the currently viewed page, and so have an equal

chance of being predicted.

Table 6.4 shows results of the experiments. One can notice that semantic-rich 1st-order Markov models have the same model size as regular 1st-order models. This is because no pruning is used in semantic-rich models. While these semantic-rich models solve the problem of contradicting prediction, they also provide accuracy very close to that of regular 1st-order models. This accuracy differs depending on the nature of the sessions in the data sets. For example, in *DS-3*, the accuracy of semantic-rich model (that is 29.80%) is about equal to that of non semantic-rich (which is 30.02%), because, once the web log was examined, it was found that users traversed pages that are highly semantically related and would resemble the structural relations between the pages in the web site.

Table 6.4: Comparing accuracy and model size of different Markov models for the different data sets. Sem. 1st-order stands for semantic-rich 1st-Order Markov models, and FPSMM stands for frequency-pruned selective markov models with 0 frequency threshold.

Model	DS-1		DS-2		DS-3	
	Accuracy in %	Size	Accuracy in %	Size	Accuracy in %	Size
1st-order	17.50	870	17.50	6320	30.02	992
Sem. 1st-order	12.50	870	19.02	6320	29.80	992
2nd-order	18.00	25230	18.70	181858	30.12	30752
All$K$th-order	**26.34**	757770	19.80	1526176	29.52	985056
FPSMM	25.81	21547	**25.03**	807617	**31.83**	12741

Highlighted in boldface in Table 6.4, are the highest prediction accuracies for each data set, which show that frequency-pruned SMM is mostly the best choice for high accuracy and small model size, as the results show an average decrease of 57% in model size, with only an average of 2.7 deviation in accuracy. The second best is the 2nd-order markov model, which confirms the findings in Deshpande and Karypis [2004]. But, could there be a better compromise in which the accuracy is higher and the model size is smaller? In an attempt to answer this, the proposed semantic-pruned SMMs are built with differing values for η, shown in Table 6.5.

It can be noted from Table 6.5 that, as the value of η decreases, so does the model size. This is expected, since less η means more pruning will take place, and accuracy also decreases for the same reason. At a value of $\eta = 70$, the semantic-pruned SMM does provide an accuracy close to that of its respective frequency-pruned SMM (FPSMM) in Table 6.4, with an average difference of only 1.91 percentage points in accuracy, and at

Table 6.5: The accuracy and space size of the proposed model.

Max Semantic	DS-1		DS-2		DS-3	
	Accuracy		Accuracy		Accuracy	
Distance η	in %	Size	in %	Size	in %	Size
$\eta = 5$	5	2697	0	31163	11.71	5032
$\eta = 20$	10.71	7283	12.9	89039	20.00	9118
$\eta = 50$	20.01	10066	19.35	254399	27.43	10230
$\eta = 70$	**22.77**	**15844**	**23.17**	**726855**	**31.00**	**11420**
$\eta = 90$	24.95	20364	24.81	755641	31.34	12420
$\eta = 110$	25.81	21547	25.00	774275	31.80	12741

the same time it does maintain a smaller state space, with an average of 16% decrease in model size.

6.6.3 Results with an Alternate Combination Function

To test the effect of the combination function $\oplus$ on the accuracy and size of the generated Markov models, an alternate function is used that will scale the transition probability by the amount of the semantic distance, as follows:

$$w_{o_i,o_j} = Prob(\mathcal{P}_{x+1}|\mathcal{P}_x) \times \begin{cases} 1 + (1 - \dfrac{\frac{\gamma_{o_i,o_j}}{n}}{\sum\limits_{k=1} \gamma_{o_i,o_k}}) & , \ \gamma_{o_i,o_j} > 0 \\ \\ 1 & , \ \gamma_{o_i,o_j} = 0 \end{cases} \tag{6.5}$$

In Table 6.6 we report results of using a semantic-rich and semantic-pruned Selective Markov model with $k = 3$ and varying values of η, and compare results with semantic-rich frequency-pruned Selective Markov model (SR-FPSMM), the last row in Table 6.6.

Highlighted in the table are optimal values of η for each experimental dataset where the accuracy is at its best in regards with the model size. In these tests the semantic distance matrix was populated with random values from a normal distribution with mean=50 and variance=40. It can be observed that optimal results are obtained with $\eta = 60$, which is a value around the average, as noted before in experiments of Table 6.5. The experiment also shows that semantic-rich and pruned SMM have a minor increase in accuracy over semantic-pruned SMM (compare highlighted rows of Tables 6.5 and 6.6).

Table 6.6: The accuracy and space size of a semantic-rich and pruned Selective Markov model with $k = 3$ and varying values of η as compared to a semantic-rich frequency-pruned Selective Markov model of the same order (compare values in Boldface).

Max Semantic Distance η	DS-1 Accuracy in %	Size	DS-2 Accuracy in %	Size	DS-3 Accuracy in %	Size
$\eta = 10$	8.3	3682	7	51371	16.24	7500
$\eta = 20$	11.60	7283	14.8	89039	22.31	9118
$\eta = 50$	22.81	10066	21.5	254399	27.33	10230
$\eta = 60$	**25.75**	**15844**	**24.57**	**726855**	**32.14**	**11420**
$\eta = 90$	25.80	20364	24.50	755641	32.25	12420
SR-FPSMM	**25.15**	**21547**	**27.13**	**807617**	**33.38**	**12741**

6.7 Summary

The availability of semantic information and the tradeoff problem between state space complexity and accuracy in Markov models [Pitkow and Pirolli, 1999], trigger a need to integrate this information in the mining process. The proposed integration into the transition probability matrix of lower order Markov models is prsented as a solution to the tradeoff problem, resulting in semantic-rich lower order Markov models, and solving the problem of *contradicting predictions*. We also propose to use semantic information as a criteria for pruning states in higher order (where $k > 2$) Selective Markov models [Deshpande and Karypis, 2004], and compare the accuracy and model size of this idea with semantic-rich markov models and with traditional Markov models used in the literature. It is found that semantic-pruned SMM have a smaller state space size than frequency-pruned SMM and provide nearly an equal accuracy, with a possibility to increase this accuracy if the model is enriched with semantics.

When a semantic-rich Markov model is built and the weight matrix **W** is produced, a suggestion is to build an uncertainity matrix that holds the error margins during prediction. In this case, when a prediction is made and found to be incorrect, the weight matrices should be updated with this information and the weights (or probabilities in a transition probability matrix) are adjusted in a process of Reinforcement Learning. This means that learning can take place online with prediction, requiring scalable online performance. An advanced model that fits this scenario is that of an Artificial Neural Network, which is left for future work. For an introduction and survey on Artificial Neural Networks I refer the reader to Chapters 1 and 2 of my Master's thesis [Mabroukeh, 1998].

Chapter 7

Semantics-aware Web Recommendation

In the third phase of SemAware, the mined frequent patterns (now in the form of frequent semantic objects) and the Markov models built in the previous phase, are used for generating recommendation in the form of semantics-aware association rules, next item prediction and top-n recommendations. This chapter has two parts. First we describe recommendation from association rules combined with Markov predictions, then we propose an innovative top-n Ontology-based WRS that overcomes problems of scalability, cold start, and content overspecialization. This work is accepted for publication in DESWeb 2011 (a workshop of the IEEE ICDE 2011 conference) [Mabroukeh and Ezeife, 2011], and is partially published in Mabroukeh and Ezeife [2009b].

7.1 Recommendation from Mined Frequent Patterns

Frequent semantic objects are used in semantics-aware association rules for product and page view recommendation. The importance of these rules lies in the ability to rollup the ontology hierarchy and perform concept generalization, to expand the recommendation set and provide interpretable customer behavior analysis. The results of a semantics-aware Markov model for next-page-request predictions can be used as a complement to these association rules, such that, in the case that prediction cannot be made using the Markov model, then association rules can be consulted, and vice versa.

Semantics-rich association rules provide more accurate higher quality recommendation than regular association rules. For example, consider the following two rules with the same confidence measure:

$$o_3 o_2 \rightarrow o_4$$
$$o_3 o_2 \rightarrow o_5$$

such that $\gamma_{o_2,o_5} < \gamma_{o_2,o_4}$, meaning that o_5 is semantically closer to o_2 than o_4 is. Then the recommendation engine will prefer to recommend o_5 over o_4.

In addition, concept generalization capability is not provided by regular association rules. An example of such capability is the generalization "users who rent a movie will also buy a snack", which is a taxonomic abstraction resulting from mapping the frequent sequence $<be>$ to the ontology, and looking at higher levels in the concept hierarchy for generalization (where b refers to a horror movie and e refers to a bag of potato chips).

As *support* measures the usefulness of a pattern, the *confidence* is an interestingness measure that represents the certainity of an association. Statisticaly speaking it represents a conditional probability. For example, the confidence of the association rule $a \rightarrow b$ is the conditional probability $P(b|a)$. A minimum confidence threshold is specified by the decision maker (i.e. SemAware user) to discard insignificant association rules. Nevertheless, certainity is not enough to decide if a rule makes sense or not; semantic information adds such interpretability to association rules and reveals useless ones. Example of queries that can be answered using semantics-rich rules are:

- "What accessories are associated with washing machines?"

- "What is required to buy any *kind* of a cleaning device?"

- "What *kind* of camera is usually associated with flat panel TVs?"

Previously we could not answer these queries with regular association rules, because pageviews only refered to actual HTML pages, not to concepts.

7.1.1 Examples and Experimental Results

Consider the set of frequent semantic objects $fo=\{o_1, o_2, o_5, o_1 o_2, o_1 o_5, o_2 o_1, o_2 o_2, o_2 o_5, o_1 o_2 o_5\}$, resulting from mining the sequence database of the running example in Table 4.2 (mining is traced in Example 5.1). The following set of semantics-rich association rules is generated:

$$o_1 \xrightarrow{0.8} o_2$$
$$o_1 \xrightarrow{0.8} o_5$$
$$o_2 \xrightarrow{0.6} o_1$$
$$o_2 \xrightarrow{0.8} o_5$$
$$o_1 \xrightarrow{0.6} o_2 o_5$$
$$o_1 o_2 \xrightarrow{0.75} o_5$$

The value at each arrow shows the confidence of the respective rule. Algorithm 4 provides a procedure that utilizes the underlying ontology and the proposed semantic distance measure (from Equation (4.1), page 51) to perform roll-up operations and provide a generalized semantic interpretation for any given rule, while Algorithm 5 uses the same input to perfom drill-down operations and provide a detailed semantic interpretation of the given rules. The user has the option to roll-up or drill-down on any side of the association rule and in any degree.

Algorithm 4 *An algorithm to roll-up association rules based on domain ontology*

$\boldsymbol{Ontology\_RollUp}(o_i, \mathcal{O})$:

Input: semantic object $o_i = < pg_i, ins_k >$, see def. 6 pg. 48,
  domain ontology $\mathcal{O}$,

Output: semantic concept c^{up}

Algorithm:

1: Retreive concept $c_k \in \mathcal{O}$ to which ins_k belongs, by mapping the input semantic object to the ontology

2: Find $\arg\min_{c_j}(\gamma_{c_k,c_j})$, such that $\gamma_{c_k,c_j} = 1/\dfrac{\left|\mathcal{C}_{c_k}^- \cap \mathcal{C}_{c_j}^-\right|}{\left|\mathcal{C}_{c_k}^- \cup \mathcal{C}_{c_j}^-\right|}$ (Eq. (4.1), pg. 51) and c_j is a direct subsumer of c_k, $c_k \lhd c_j$
   That is find c_j the parent of c_k that has the highest similarity (i.e. lowest semantic distance) to c_k

3: $c^{up} = c_j$

4: **return** concept c^{up}
   end

These two proposed algorithms take as input the semantic object directly, from any side of the association rule, rather than taking the whole association rule, then map the semantic object to its corresponding concept in the ontology (line 1 in both algorihms), calculate the most similar parent or son (line 2 in both algorithms for roll-up and drill-down respectively) using the proposed similarity measure (Eq. (4.1)) and subsumption relationships, then return the respective parent or son.

Algorithm 5 *An algorithm to drill-down association rules based on domain ontology*

Ontology_DrillDown$(o_i, \mathcal{O})$:

Input: semantic object $o_i = < pg_i, ins_k >$, see def. 6 pg. 48,
  domain ontology $\mathcal{O}$,

Output: semantic concept c^{down}

Algorithm:

1: Retreive concept $c_k \in \mathcal{O}$ to which ins_k belongs, by mapping the input semantic object to the ontology

2: Find $\arg\min_{c_j}(\gamma_{c_k,c_j})$, such that $\gamma_{c_k,c_j} = 1/\dfrac{\left| \mathcal{e}_{c_k}^- \cap \mathcal{e}_{c_j}^- \right|}{\left| \mathcal{e}_{c_k}^- \cup \mathcal{e}_{c_j}^- \right|}$ (Eq. (4.1), pg. 51) and c_k is a direct subsumer of c_j, $c_j \lhd c_k$

   That is find c_j the direct son of c_k that has the highest similarity (i.e. lowest semantic distance) to c_k

3: $c^{down} = c_j$

4: **return** concept c^{down}

   end

Example 7.1 (semantics-rich association rules): Consider the rule $o_2 \xrightarrow{0.8} o_5$, and consider the ontology in Figure 2.5 and Table 4.3 of semantic objects mappings. The rule can be interpreted as "I am 80% sure that a "Dry Battery" is a popular item to buy after buying a "Still Camera"". Now, by performing a one-fold drill-down on the left-hand side and a one-fold roll-up on the right-hand side of the rule in the ontology given, it can be interpreted as "I am 80% sure that any kind of "Battery" is a popular item to buy after buying a "Digital" camera". This very simple example shows the quality of semantics-rich association rules, such results are made possible by using the proposed semantics-aware SPM of SemAware and the proposed semantic distance measure. Regular association rules do not provide such interpretation and capability, they can mimic the given example, by mapping the rules to an ontology (usually a shallow one) in a tedious postprocessing phase and use only is-a relations to roll-up and drill-down the shallow ontology [Eirinaki et al., 2003; Vanzin and Becker, 2006]. It is important to mention here that the confidence value should change with roll-up and drill-down, becuase generalization and specialization are involved which affects the confidence in the resulting association rules. This is not considered in the proposed algorithms (Algorithm 4 and Algorithm 5). The proposed algorithms will take as input only a semantic object, not a complete association rule, and perform roll-up or drill-down on this single provided object. An automatic relation for computing new confidence from roll-up and drill-down operations on the complete rule is left for future work.

While some association rules share the same left-hand side and the same confidence value, their right-hand side results are different, leading to confusing recommendations. To solve this problem the resulting Markov model in SemAware is consulted, based on Algorithm 6. This is done by aligning the left-hand side of the association rule with the corresponding entry in $\mathbf{W}$ in the approriate Markov model. The semantic-rich semantic-pruned Selective Markov model resulting from the second phase of SemAware (Chapter 6) is used here.

Example 7.2 (contradicting association rules): Consider the example above of the two association rules $o_3 o_2 \to o_4$ and $o_3 o_2 \to o_5$, with the same confidence. The markov model will be consulted to find $Prob(o_4|o_3 o_2)$ and $Prob(o_5|o_3 o_2)$, by locating the row for $o_3 o_2$ in the weight matrix $\mathbf{W}$ for the 2nd-order model and comparing both probabilities. The one with the highest probability will be considered as the most certain result of the association rule. If no entry is found for $o_3 o_2$ in the model, then the 1st-order model will be consulted at the row of o_2. If still no entry is found in $\mathbf{W}$ of the 1st-order model, then we resort to the semantic distances γ_{o_2,o_5} and γ_{o_2,o_4}, and return the semantic object with the least distance. Finally, the Markov model will be updated by adding an entry in $\mathbf{W}$ for $o_3 o_2$ with either o_4 or o_5, whichever has the least semantic distance. This entry will be the value of the semantic distance combined with $\mathbf{W}$ according to the combination function $\oplus$ adopted in the second phase of SemAware.

To experiment with contradicting association rules, we went back to the experiments carried on PLWAP-sem, using the same medium data set C10T6N40S4D50K and the same large data set C15T8N100S4D100K, with min_sup of 1% and $\eta = 8$, since it is reported that this value for η is found to be optimal for these datasets with PLWAP-sem (see Section 5.4.2). Association rules with confidence less than 85% are removed. Also a 3rd-order semantic-rich and semantic-pruned SMM is built for each dataset separately. Table 7.1 reports the results of mining these sets. These results show the importance of the

Table 7.1: Experiments for association rules on different datasets using PLWAP-sem

Testing for association rules using PLWAP-sem		
	Data sets	
	|D|=50K	|D|=100K
# frequent patterns	1498	1298
# assoc. rules	575	225
# assoc. rules with same LHS and confidence	217	157
# hits in the markov model	200	142
avg. time for generating predictions from markov model	0.45 sec.	0.33 sec.
# times markov model was updated	17	15

Algorithm 6 *An algorithm to replace association rule result with result from a Selective Markov model in SemAware*

Assoc2Markov$(L, \mathbf{W})$:

Input: semantic objects in the left-hand side of the association rule, the sequence $L =< o_{i-k+1} \ldots o_i >$

  weight matrix of the semantic-rich Selective Markov Model of order k, $\mathbf{W}$

Output: next accessed object o_{i+1}

Algorithm:

1: SemAware provides a semantic-rich 3rd-order Selective Markov model as output of its second phase

2: k determines the order of the semantic-rich SMM to use

3: Align the sequence L with the markov model in order to query its $\mathbf{W}$
    (i.e. in order to find $Prob(o_{i+1}|o_{i-1}o_i)$ we need a 2nd-order model, and locate the row in $\mathbf{W}$ that represents the sequence $L =< o_{i-1}o_i >$)

4: **if** there is no entry in $\mathbf{W}$ for L **then**

5:    remove the first object from sequence L

6:    k=k-1

7:    try $(k)^{th}$-order Markov to align the new shorter sequence, and so on until 1st-order model

8: **end if**

9: **if** an entry is found in $\mathbf{W}$ for the sequence in any of the orders of the semantics-rich Selective model **then**

10:    Find $o_{i+1} = \arg\max_{p_o \in \mathbb{P}}\{Prob(\mathcal{P}_{i+1} = p_{o_{i+1}}|\mathcal{P}_i, \mathcal{P}_{i-1}, \ldots, \mathcal{P}_{i-(k+1)})\}$, the next accessed object o_{i+1} with maximum transition probability

11:    **return** o_{i+1}

12: **else**

    get o_{i+1} from association rule that has the smallest semantic distance to o_i, Add it to the kth-order Markov model where the sequence L can be aligned, **return** o_{i+1}

13:15: **end if**

    end

---

proposed system, as it was able to resolve an average 91% of the contradicting association rules using a fast mining algorithm, PLWAP-sem. These association rules constitute an average 54% of the generated association rules. The average time to consult the markov models (pre-built during the second phase of SemAware) to resolve for one association rule is 0.39 seconds which is very fast and suitable for commercial production.

7.2 Ontology-based Recommendation from User-Provided Tags

Due to the continuous development and use of Web 2.0 technologies, web applications are starting to enable their users to tag items, adding a form of semantic value to them. An ontology-based WRS is defined in this thesis (Definition 5, page 29) as a system that maximizes the utility function $\lambda(u, p_i, \mathcal{O})$ to find the top-n items of interest to a user. In a model where the web pages are not annotated with ontology concepts, it is desirable to make use of these user-provided tags as semantic information that describes the tagged items. Such systems store user-provided tags in the database along with item description and related information.

Given a database of items and their associated tags, the web log can be represented as a set $W = \{< p_i, T_{p_i} >: 1 \leq i \leq |W|\}$, containing pairs of item p_i and set of tags T_{p_i}, where $T_{p_i} = \{g_j : 1 \leq j \leq |\mathbb{G}|\}$, g_j is a tag and $\mathbb{G}$ is the set of all user-provided tags. Following is an example web log:

$$\begin{aligned}
\mathcal{W} = \{ \; &< p_2, [\text{``close''}, \text{``quality''}, \text{``clear''}, \text{``lens''}] >, \\
&< p_5, [\text{``tight''}, \text{``water''}, \text{``ocean''}, \text{``dive''}] >, \\
&< p_3, [\text{``scuba''}, \text{``mask''}, \text{``snorkel''}, \text{``dive''}] >, \\
&< p_1, [\text{``professional''}, \text{``video''}, \text{``flash''}, \text{``digital''}] >, \\
&< p_4, [\text{``dry''}, \text{``toy''}, \text{``small''}] >\}
\end{aligned}$$

The active user u provides a set of tags, either directly as a query or indirectly by providing these tags as keywords of her favorite product features, stored in her profile when she signs up in the web site.

7.2.1 Preprocessing and Clickstream Mapping

Since tags are freely provided by users, there is no limit on their number, which makes them inapproriate to use as dimensions in a Vector Space Model. To avoid any dimensionality problems and to avoid the use of time-consuming clustering methods, we propose to map the tags to the concepts of the underlying domain ontology. Section 7.2.4 proposes a novel method for more dimensionality reduction using the domain ontology to solve the sparsity and scalability problems of a WRS.

To map tags to their respective ontology concepts WordNet[1] [Miller et al., 1990] is used

---

[1]WordNet is a lexical database containing English nouns, verbs, adjectives and adverbs organized

as a thesaurus, to find the similarity between an item and all the tags associated with it. In detail, each tag in p_i is compared against each concept in the ontology. This is done by computing the similarity between the tag and the concept, both the tag and the concept are located as two words in WordNet (say n_1 representing concept c_j, and n_2 representing a tag g from the set T_{p_i}), then Wu and Palmer similarity measure [Wu and Palmer, 1994] is computed as

$$sim(n_1, n_2) = \frac{2 \times D_3}{D_1 + D_2 + 2 \times D_3} \tag{7.1}$$

where n_3 is the Lowest Common Ancestor of n_1 and n_2, D_1 is the distance (in number of nodes on the path) between n_1 and n_3, and D_2 is the distance between n_2 and n_3, while D_3 is the distance between n_3 and the root.

After users tag items, the SemAware preprocessor cleans these tags before storing them in the database. The cleaning process goes through the following steps:

Tags clean-up: Non relevant and meaningless tags are removed from each set T_{p_i}, e.g., stop words (frequently occurring insignificant words like "the"), words with numbers, and numbers. Clean tags will be treated as nouns when mapped to WordNet.

Word Stemming: Words are reduced to their root form using the standard Porter stemming algorithm [Porter, 1980], which removes the common morphological and inflexional endings of words. For a group of similar tags, the shortest form is chosen and is the one that is mapped to WordNet.

After cleaning, each tag (now treated as a word) is mapped to its proper *synset* in WordNet. A synset is the smallest unit in WordNet; it represents a specific meaning of a word, and it also includes its synonyms. Each synset has a *gloss* that defines the concept it represents. For example, the words "night", "nighttime", and "dark" constitute a single synset that has the following gloss: "the time after sunset and before sunrise while it is dark outside". Synsets are interconnected using is-a-kind-of and is-a-part-of hierarchies as semantic relations [Simpson and Dao, 2010]. To compute the similarity between two words, the semantic similarity between their word senses is calculated using Wu and Palmer measure as described in Equation (7.1) via hyponym/hypernym relations (is-a relations). Due to the limitation of is-a hierarchies only "noun-noun" relations are considered.

Eventually, each item p_i can be represented as a vector $\vec{p_i}$ of similarity scores over the space of ontology concepts, in which each entry represents the average similarity score of

---

into synonym sets, each representing one underlying lexical concept. It provides mechanisms for finding synonyms, hypernyms, hyponyms, etc. for a given word.

all the tags associated with the item p_i with respect to one concept. Thus, resulting in a matrix of items-concepts correlations, called the PC matrix, similar to Figure 7.1.

Definition 11. *The items×concepts matrix is the matrix $PC_{I \times J} = \{sim(T_{p_i}, c_j) : I \leq |\mathbb{P}|, J \leq |\mathcal{C}|\}$, where $sim(T_{p_i}, c_j)$ is the similarity score between the set of tags T_{p_i} for item p_i and the concept c_j, $1 \leq i \leq |\mathcal{W}|, 1 \leq j \leq |\mathcal{C}|$, where $\mathbb{P}$ is the set of all items in the database.* □

Example 7.3 (*PC* **matrix**): Follow with *Clickstream_mapping* procedure in Algorithm 7, and assume that the concepts $c_1 = $ "*camera*", $c_2 = $ "*lens*", $c_3 = $ "*battery*", $c_4 = $ "*dive*". In lines 1-3 of the algorithm and to compute the similarity between p_2, from W and c_1, we use Wu and Palmer measure with WordNet as described above, to get the similarity score between each tag associated with p_2 (the tags 'close', 'quality', 'clear', 'lens') and the concept 'camera'. The result will be 0.346, 0.354, 0.206, 0.517 respectively. The average score of 0.356 (Line 3 of the algorithm) is used to represent the overall similarity score $sim(T_{p_2}, c_1)$. By computing the similarity score for each item with each concept in this example, we get the matrix in Figure 7.1 (Lines 4-7).

$$PC = \begin{bmatrix} & c_1 & c_2 & c_3 & c_4 \\ p_1 & 0.187 & 0.256 & 0.202 & 0.113 \\ p_2 & 0.356 & 0.463 & 0.232 & 0.202 \\ p_3 & 0.156 & 0.116 & 0.098 & 0.491 \\ p_4 & 0.213 & 0.165 & 0.111 & 0.129 \\ p_5 & 0.213 & 0.173 & 0.113 & 0.403 \end{bmatrix}$$

Figure 7.1: Example of *PC* matrix with item-concept similarity scores.

The PC matrix can get large and sparse, for this we propose a dimensionality reduction method that utilizes the hierarchy of the ontology, below in Section 7.2.4.

7.2.2 Recommendation to the Active User

Tags are provided by the active user u. Let's call the user profile, the *active page p^u*. These tags T_{p^u} are then mapped to ontology concepts, in a way similar to the offline clickstream mapping process described in Section 7.2. So, p^u can now be modeled as a vector of similarity scores in the space of concepts, $\vec{p^u} = [s_1^u \ s_2^u \ldots \ s_k^u]$, where $k = |\mathcal{C}|$ and $s_i^u = sim(T_{p^u}, c_j)$. For the adopted example, follow with procedure *Online_Recommend* in Algorithm 7 and assume $T_{p^u} = \{$"*camera*", "*digital*", "*zoom*"$\}$, one can compute $\vec{p^u} = [0.400\ 0.234\ 0.266\ 0.220]$ from Lines 2-3 of the procedure. To find the top-n recommendations (Lines 4-6), $\vec{p^u}$ is matched against every row in PC to find their relatedness, using

cosine similarity,

$$rel(\vec{p^u}, \vec{p_i}) = \frac{\vec{p^u} \cdot \vec{p_i}}{\left\|\vec{p^u}\right\| \left\|\vec{p_i}\right\|} \tag{7.2}$$

such that $1 \leq i \leq I$, I from Definition 11 is the number of rows in the PC matrix.

The results are ranked in descending order and the items with top-n relatedness scores are added to the recommendation set $\mathcal{S}$ (Lines 7-8 of procedure *Online_Recommend*), i.e.,

$$\mathcal{S} = \arg \max_{p_i \in \mathbb{P}}^{n} (rel(\vec{p^u}, \vec{p_i})) \tag{7.3}$$

Applying this to the example here, requires that $\vec{p^u}$ be matched with every row of the matrix in Figure 7.1, using equations (7.2) and (7.3). If $n = 3$, the recommendation set will be $\mathcal{S} = \{p_4 : 0.987,\ p_1 : 0.940,\ p_2 : 0.936\}$, where the score associated with each recommended item is the score from equation (7.2).

7.2.3 Quality of Tags and Tag Spam

Although the cleaning process does remove a number of unwanted tags and meaningless words, it could happen that a user overtags an item or a group of items, for several purposes like promotion or spamming. A user is limited to provide only a certain number of tags per item, this should solve item overtagging problem. To solve the problem of spamming[1], tags can be assigned frequency scores using TF-IDF (term frequency $\times$ inverse document frequency), and by treating the set of tags of one item T_{p_i} as a document, and one tag as a term. TF-IDF assigns a weight to a term in a document that is low when the term occurs several times in a document and lowest when the term occurs in almost all documents. In this case tags that have low scores are considered insignificant and can be removed during preprocessing.

7.2.4 Using the Ontology Hierarchy for Dimensionality Reduction

Matrix sparsity is an important problem in recommendation systems that contributes to the scalability problem. In the proposed model, a good number of items might be highly similar to only a subset of the concepts. Also, another reason to reduce the dimensionality is to decrease the time required to generate a recommendation. The proposed method makes use of the *is-a* hierarchy of the ontology, in which two or more leaf concepts, represented by their respective columns in the PC matrix, are combined by their Lowest Common

---

[1]We can define spamming in this context as the process of applying a certain tag to a large number of items or a single item, by a single or multiple users. This also applies on tags of the same meaning being applied repeatedly to a single item by a single user

Ancestor (LCA) into one column. A basic combination formula using a weighted average is used, in which the distance from each leaf concept to the LCA is used as a decay factor of the similarity score.

$$sim(p_i, c_{LCA}) = \frac{\sum_j \frac{1}{d_j} sim(p_i, c_j)}{j} \qquad (7.4)$$

such that c_j is a leaf node in $\mathcal{O}$ and $sim(p_i, c_j) > 0$, where c_j is subsumed by c_{LCA} denoted as $c_j \triangleleft c_{LCA}$ and d_j is the number of edges between c_j and its ancestor c_{LCA}. The result of this formula is the similarity score between the LCA and each row in the PC matrix p_i, this formula is applied for each c_{LCA} For example, if concepts c_2 and c_3 in Figure 7.1 both have c_7 as their LCA, then the two columns representing these concepts in the figure, can be combined by equation (7.4) into one column representing c_7 and the matrix size is thus reduced by one column. If every two columns in the PC matrix are reduced using this method to one column, then the reduced matrix will have a minimum number of columns half of that of the original PC matrix.

Another way to reduce the dimensionality of the PC matrix is to use *Feature Subset Selection* (FSS), by removing columns of concepts that have the lowest similarity scores. Later in Section 7.3, we experiment with FSS and compare results with the proposed LCA-based reduction.

7.2.5 Expanding the Recommendation Set

An important problem in recommendation systems is that of content overspecialization. As a WRS provides top-n highly related items, these item are not of enough variety to the user. For example, if a user shows interest in a narrow topic, the recommended items will be from the same topic, especially if the user does not have a considerable browsing/purchase history, which is not good for customer retention and it leaves only a small possiblity for new items to surface. The use of domain ontology plays an important role here, giving SemAware a huge advantage over recommender systems that do not use domain knowledge, in that the recommendation set can be expanded by incorporating items from other concepts that are related to the ones represented in $\vec{p_u}$. A full ontology plays a better role here than a shallow ontology, in the sense that all relations in the ontology are utilized to provide a recommendation, in the process of *Spreading Activation* (Procedure *Expand_RecSet* in Algorithm 7). The process starts from the concept c_j to which $\vec{p^u}$ has the highest similarity (Lines 1-2 of the procedure). Then, the recommendation is spread over all relations from this concept to other concepts (the set of relations $\mathcal{R}^{c_j}$), generating recommendations of items belonging to those concepts. The relations are first ranked by

their relation hierarchy score $\psi_{r_{jk}}$ (Def. 4), which is a score assigned to each relation by the ontology engineer or the ontology learning algorithm that characterizes the relation hierarchy $\leq_{\mathcal{R}}$ (Lines 3-8). This is not possible in a shallow ontology, because only *is-a* relations are present. For example, if a user is recommended a video camera product, to expand this recommendation (Lines 4-8), the system will utilize the relations (follow with ontology in Figure 2.5):

$$requires(``VideoCamera", ``VideoFilm")$$
$$is - a(``VideoCamera", ``Camera")$$
$$has - a(``Camera", ``Lens")$$
$$is - a(``VideoCamera", ``Camera") \;\wedge\; has - a(``Camera", ``Lens")$$
$$\Rightarrow has - a(``VideoCamera", ``Lens")$$

So, object instances of "Lens" and "Video Film" concepts are generated and added to the recommendation set. Such relations, like *has-a* and *requires*, are not present in a shallow ontology, and such recommendations can not be generated from simple association rules if only sequential pattern mining is employed by the WRS.

Algorithm 7 in page 95 provides step-by-step details of all the procedures discussed for top-n Ontology-based WRS (called SemAwareIN).

7.2.6 Computational Complexity

The complexity of the proposed top-n WRS depends on the amount of time required to build the offline model and the amount of time required to compute the active recommendation online based on this model. While building the model we compute the similarity of each item p_i to each ontology concept c_k for building the PC matrix. If there are n items and m concepts, then the upper bound on the complexity of this step is $O(mn)$. When LCA-based dimensionality reduction is used, the number of m concepts is reduced to $k < m$, making the complexity $O(kn)$. If every two columns in the PC matrix are reduced to one column then $k = m/2$.

The time complexity for computing active recommendations is linear and proportional to the number of items n in the PC matrix, $O(n)$, since p^u is compared against each item in PC. Thus, the overall complexity of the proposed system is $O(mn) + O(n) = O(mn)$.

7.3 Experimental Evaluation

7.3.1 Methodology

For experimental evaluation, we use the MovieLens dataset[1]. The proposed system (called *SemAwareIN*) is built using C#, and all experiments are performed on a 1.87GHz Intel Core Duo machine with 2GBytes of memory, with the MO-Movie Ontology [Bouza, 2010] as the underlying ontology, and WordNet 2.1[2] used for similarity computations. The MovieLens dataset contains 100,000 tags for 10,681 movies, made by 71,567 users. The tags are cleaned and prepared as described in Section 7.2.1 by removing non-words, stop words, and keywords with numbers and by word stemming. The number of clean tags extracted from the dataset is 82,454 tags.

The experiments are conducted using 5-fold cross-validation method to test the accuracy of the proposed system, as error metrics methodologies (like RMSE) are not a natural fit for evaluating top-n recommendation systems [Cremonesi et al., 2010]. The MovieLens dataset is divided into five mutually exclusive sets, and at each time four of these sets are used for training (constituting 80% of the origial dataset) while the remaining set (20%) is used for testing. At every fold of the cross-validation, the PC matrix is built, such that the items represent movie IDs and the concepts represent leaf concepts in the MO ontology. The tags T_{p^u} from each row in the test set are used as input to the recommendation system. They are mapped to the ontology concepts and the vector $\vec{p^u}$ is created as described in Section 7.2.2. Then, the recommendation set $\mathcal{S}$ is generated using Equation (7.3) with $n=3$, and each recommended movie is matched against the movie in the current test row from the test set. If the movie title is the same or the movies are of the same ontology concept, then that is considered a *hit*. Otherwise, it is a *miss*. *Accuracy* is measured at each fold of cross-validation, as the percentage of hits from the total number of trials made on the test set. The average accuracy is finally computed over all of the five folds.

7.3.2 Results

After running the 5-fold cross-validation test described, the average accuracy was found to be 82%, which is very good. To confirm this we compare SemAwareIN with non-ontology-based top-of-the-art item-based recommendation algorithms, namely TopPop (which recommends top-n items with highest popularity) and NNCosNgbr (which uses kNN clustering with item-to-item similarity to recommend top-n items). For details of these two

---

[1]available at http://www.grouplens.org/node/73
[2]available at http://wordnet.princeton.edu/wordnet/download/

algorithms see Cremonesi et al. [2010]. Comparisons are made in terms of *recall-at-n*, which is computed as the ratio of hits to the size of the test set at different values of n, and *precision*, which is computed by dividing recall-at-n by n. Figure 7.2 shows the recall-at-n comparisons for values of n between 5 and 20, which tell that SemAwareIN exceeds Top-Pop and NNCosNgbr in terms of recall as it reaches, at n=10, a recall of 0.61, compared to 0.28 and 0.45 for TopPop and NNCosNgbr, repectively. This means that about 61% of the top-10 recommended movies are hits (i.e., are the same as, or sematically match, the expected result). The graph in Figure 7.2 also shows error bars of 5% for SemAwareIN and its variations, and an error percentage of 10% for TopPop and 8% for NNCosNgbr, these are erros that we found during the multiple runs of these algorithms. Figure 7.3 confirms
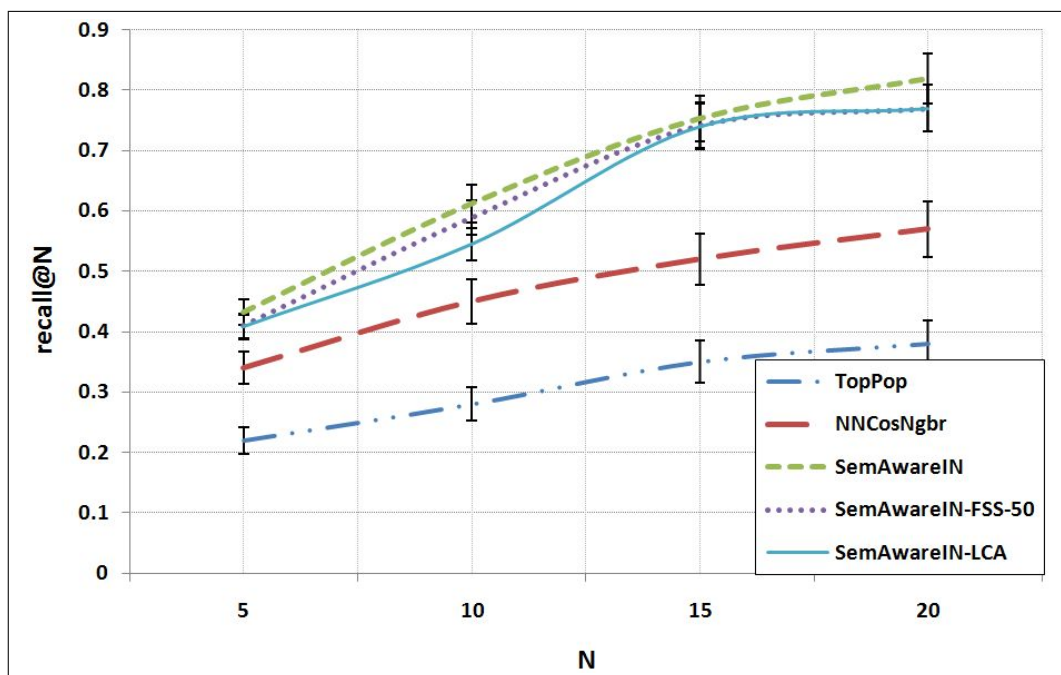

Figure 7.2: Recall-at-n comparison.

that the proposed algorithm outperforms the two popular algorithms TopPop and NNCos-Ngbr in terms of precision metrics. Each line in the figure represents the precision of the algorithm at a given recall.

To test the effect of dimensionality reduction on solving the sparsity problem of the PC matrix, 5-fold cross-validation is again performed, once using Feature Subset Selection (FSS) and another time using the proposed LCA-based reduction method. In FSS 50% of the concepts are removed, those are the 50% columns in the PC matrix that hold the lowest average similarity scores. In LCA-based, leaf concepts are joined by their LCA
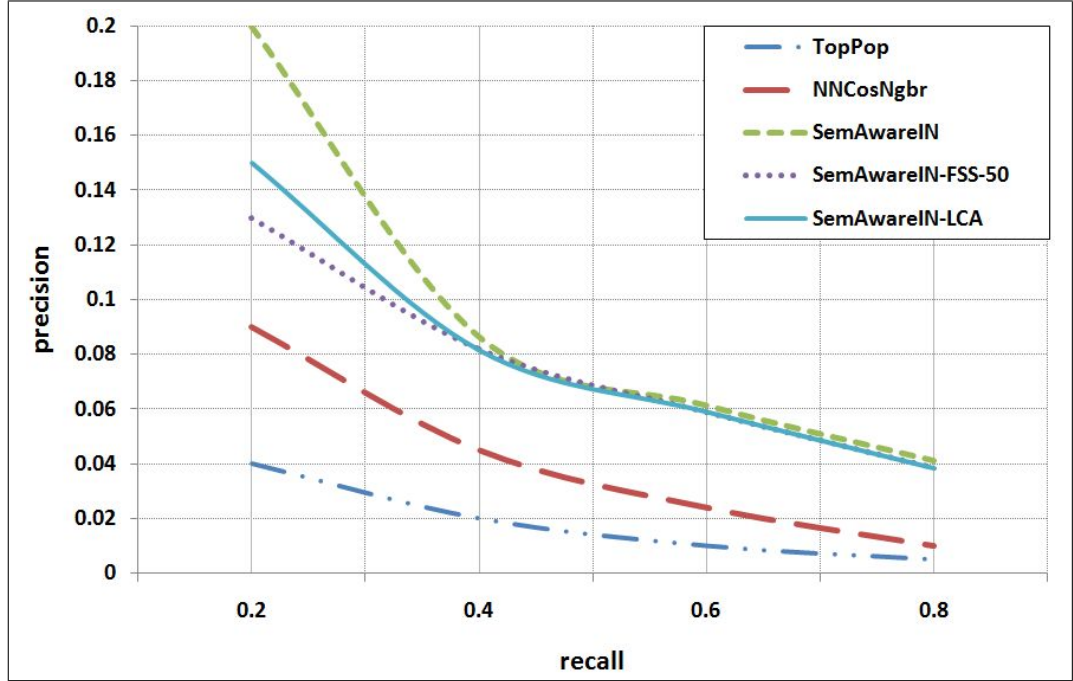
Figure 7.3: Precision vs. recall for comparison between algorithms.

as discussed in Section 7.2.4, reducing the size of the matrix in half. The FSS variation is called *SemAwareIN-FSS-50* and the LCA-based variation is called *SemAwareIN-LCA*. Figure 7.4 shows that FSS and LCA-based reductions do not compromise the accuracy and recall of the proposed algorithm, as only 1%-6% drop in accuracy is observed, which is logical in SemAwareIN-FSS-50 since only low-scoring concepts are removed, and is also encouraging in SemAwareIN-LCA, because it shows that LCA-based reduction that depends on the ontology provides a comparable performance to an algorithm with no dimensionality reduction. A close look at Figure 7.4 and Figure 7.2 reveals that SemAwareIN-FSS-50 and SemAwareIN-LCA have nearly the same performance at $n{\geq}15$, which is very close to SemAwareIN's performance without any dimensinoality reduction as comfirmed by the precision metrics in Figure 7.3.

To test the performance scalability of SemAwareIN we conducted cross-validation on datasets of different sizes (extracted from the big MovieLens dataset), as shown in Table 7.2. The table shows that the recommendation time scales very well with increasing number of tags used in training. A careful investigation of the numbers will show a linear relation. That is a good indicator, in both cases, with and without using LCA-based reduction. It is worth mentioning here that the average recommendation time using SemAwareIN-LCA is reduced by 22%, because the size of the *PC* matrix is reduced and so is the number of
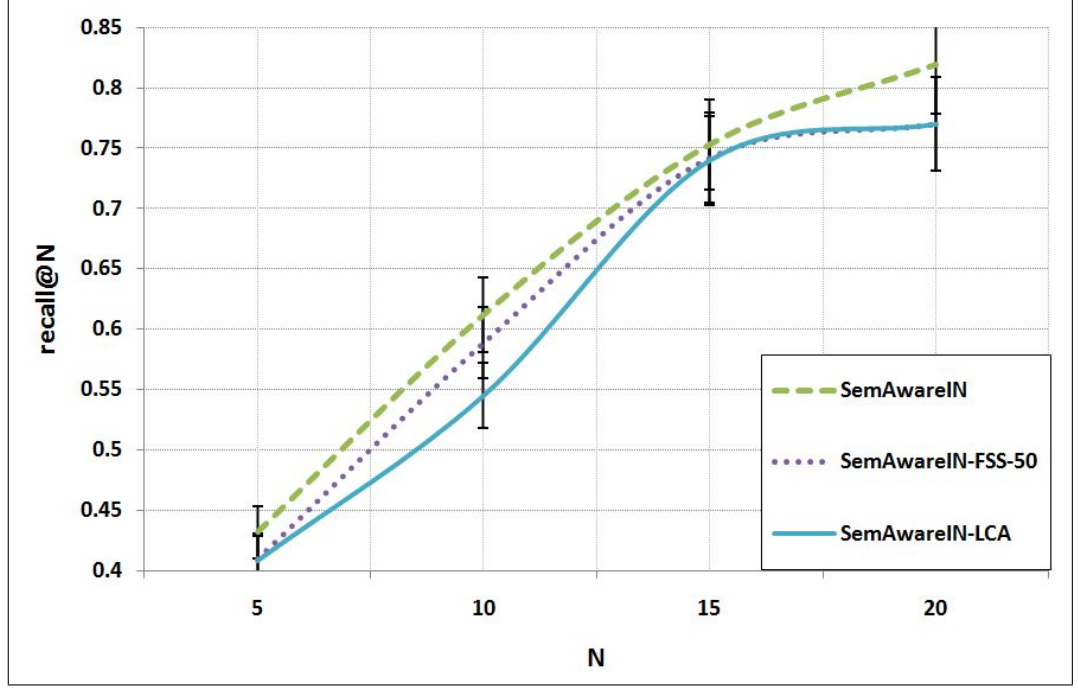
Figure 7.4: Recall-at-n for proposed algorithms.

Table 7.2: Datasets extracted from original MovieLens dataset.

Dataset Size	num of Movies	num of Tags	Avg. Rec. time (in seconds)
Small	4,739	11,948	0.973
Medium	7,308	23,895	1.117
Large	9,102	47,790	1.293

comparison operations. On the other hand, Table 7.3 shows the average accuracy versus the dataset size which is consistent with our analysis of Figure 7.4, in that SemAwareIN-LCA does not compromise accuracy, regardless of the dataset size.

The ability of the proposed system to expand the recommendation set based on ontology axioms is tested with $n=5$. In this case, *SemAwareIN-Ex* is implemented as a variation of SemAwareIN in which the recommendation set of the top-5 items is expanded using spreading activation which adds 10 more items, resulting in a set of 15 items. Recall of SemAwareIN-Ex is compared with the recall of SemAwareIN at $n=15$, and found to be 0.862, that is far better than that of SemAwareIN (which is 0.753 at $n=15$).

Table 7.3: Accuracy of proposed algorithms with different dataset sizes.

Dataset	Accuracy (in %)	
Size	SemAwareIN	SemAwareIN-LCA
Small	77.00	75.61
Medium	73.00	71.00
Large	70.52	70.30

7.4 Summary and Discussion

Recommendation in SemAware depends on semantic-rich association rules as well as probabilistic Markov models. These two results are combined in an ensemble to provide accurate informed recommendations, with the capability to make concept generalizations based on the domain ontology.

SemAwareIN is an Ontology-based WRS that we propose to provide top-n recommendation from user-provided tags. These tags are mapped to concepts of the domain ontology. Similarity measures are used during mapping and a matrix of items-concepts correlation is built offline, and is used later for online top-n recommendation. This system outperforms popular algorithms like TopPop and NNCosNgbr. A novel dimensionality reduction technique is proposed to solve the sparsity problem and reduce the size of the matrix, that depends on the hierarchy of the ontology. This technique does not compromise the accuracy of the proposed system. We also show how the recommendation set can be expanded using Spreading Activation over the ontology, taking into consideration the several available relations, which, when tested, actually raised the accuracy of the proposed model.

This work tries to answer Research Questions 3 and 4 by showing the impact of domain ontology on the recommendation results. There are two benefits in using an ontology over clustering of user tags. First, this will save the costly step of clustering, and second, a full ontology has a far better reasoning power than a topic taxonomy. In a full ontology there are several semantic relations that can be taken into consideration (as opposed to only *is-a* relation in a topic taxonomy) to provide better relatedness measures, and better interpretability.

Algorithm 7 *SemAwareIN: Ontology-based WRS with user-provided Tags*

Clickstream_mapping$(W, \mathcal{O})$:

Input: item database W,

  domain ontology $\mathcal{O}$,

Output: matrix of items×concepts PC

Algorithm:

  1: **for** each pair <item p_i, its tags T_{p_i} > in W **do**
  2:   **for** each concept c_j in $\mathcal{O}$ **do**
  3:     calculate $sim(T_{p_i}, c_j)$, which is the average similarity between each tag in T_{p_i} and c_j
  4:     store in PC, $PC[i,j] = sim(T_{p_i}, c_j)$
  5:   **end for**
  6: **end for**
  7: **return** PC
    end

Online_Recommend(PC, T_{p^u}, n):

Input: matrix of items×concepts PC,

  active user tags T_{p^u},

  number of highly similar items required n,

Output: set of top n recommended items, $\mathcal{S}$

Algorithm:

  1: [**optional**] Reduce dimensionality of PC matrix as described in section 7.2.4
  2: Map user tags to concepts similar to steps 2-3 in ***Clickstream_mapping***()
  3: store the mapping result as vector $\vec{p^u}$
  4: **for** each item p_i in PC **do**
  5:   Find $rel(\vec{p^u}, \vec{p_i})$, which is the relatedness of the item to user vector $\vec{p^u}$, using equation (7.2)
  6: **end for**
  7: Sort results from previous step and store top n results in $\mathcal{S}$
  8: **return** $\mathcal{S}$
    end

Expand_RecSet$(\mathcal{O}, \mathcal{S})$:

Input: domain ontology $\mathcal{O}$,

  set of top n recommended items, $\mathcal{S}$

Output: extended set, $\mathcal{S}^+$

Algorithm:

  1: retrieve $\vec{p^u}$ and find concept c_j to which p^u is highly similar
  2: start at concept c_j in ontology $\mathcal{O}$
  3: rank relations at c_j according to $\psi_{r_{c_j c_k}}$ that assigns scores to relation hierarchy $\leq_{\mathcal{R}^{c_j}}$
  4: **for** each relation $r_{c_j c_k} \in \mathcal{R}^{c_j}$ connecting c_j with another concept c_k **do**
  5:   get concept c_k
  6:   retrieve items related to c_k
  7:   add items to $\mathcal{S}$, to get $\mathcal{S}^+$
  8: **end for**
  9: **return** $\mathcal{S}^+$
    end

Chapter 8

Conclusions and Future Work

8.1 Conclusions

May be the biggest contribution of data mining to the world wide web, is the introduction of Web Recommendation Systems. These systems use mining methods to provide user behavior analysis to the decision maker, and product recommendations to the end user. WRS's went from being simple tools, on web sites like Amazon.com, to being an integral usage analysis tools of web server software, and customer relationship management (CRM) suites.

Surveys in this thesis point out several important problems in sequential pattern mining (SPM) algorithms and WRS's, and reveal the fact that a domain ontology with relations and axioms is far from being used. Ideas are simply borrowed from the Semantic Web, but the full power of this web, is yet to be utilized. We propose SemAware, a comprehensive ontology-based system for web usage mining and recommendation that covers the three phases of web log pre-processing, SPM with probabilistic pattern discovery and post-processing with recommendations. The use of a domain ontology enables us to solve scalability problems in SPM. It is shown that semantic-based SPM algorithms, like PLWAP-sem, can mine the web log more efficiently and effectively than regular non-semantic-based SPM and without compromising the result. The thesis also shows how semantic information drawn from the underlying domain ontology can solve the contradicting predictions problem in Markov models used for page prefetching and reduce space complexity of Selective Markov Models (SMM) used in the same domain.

This thesis carries on to answer four important research questions regarding the use of domain ontology for solving WRSs' problems and its effect on recommendation results. We propose algorithm Assoc2Markov that combines semantic-rich association rules and

semantic-rich predictive Markov models to provide more accurate recommendations, and algorithms that enable interpretation of results at different levels of abstraction, by rolling up and drilling down on ontology concepts. We also propose an ontology-based WRS that provides top-n recommendations from user-provided tags (called SemAwareIN), and show how to solve the problems of cold start with the use of tags, the problem of sparsity and scalability with a proposed novel dimensionality reduction approach, and the problem of content overspecializatin using Spreading Activation in the domain ontology. The propsed SemAwareIN outperforms popular algorithms like TopPop and NNCosNgbr in terms of precision and accuracy.

It can be concluded that the use of domain ontology in SPM and WRS's as hypothesized and made possible by this thesis does enhance and increase the effectiveness of these systems, and does provide easy solutions to several problems in them, as we have established through experiments.

8.2   Future Work

Following is a list of possible future work directions and recommendations that can emerge from this thesis, with some implementation ideas.

8.2.1   Ontology Learning Using SPM

The simple algortihm provided for building an ontology from the web log (Algorithm 1, pg. 46) requires further research and testing. Future work includes the identification of relation hierarchy by using hierarchical clustering methods, and the promotion/demotion of relations in the learned ontology according to the support or interestingness of the frequent patterns mined from the web log. In addition, methods have to be studied to better identify and label concepts as the ontology is being built.

8.2.2   Onotology Update to Match User Behavior

The results presented in this thesis are encouraging as they show the effect of using a domain ontology on SPM algorithms and the effectiveness of semantic-aware algorithms. On the other hand, scholars have debated that the ontology reflects the domain expert's view of the system and does not really reflect the actual user behavior. The main reason for me doing this research was the belief that user behavior is not totally random, but is actualy drived by logical relations among the items/web pages in the web site. Further work is recommended on ontology update. That is, the ability of SemAware to restructure

the ontology based on frequent patterns and association rules resulting from non-semantic-aware mining of the web log, that will reflect the user behavior directly into the ontology.

8.2.3   The Effect of Frequent Pattern Size

While most sequential pattern mining algorithms use the item support as the main interestingness measure for pruning out candidate sequences, the size of the potential frequent pattern is being ignored. It is believed that the size of a frequent pattern has an important effect on the decision maker and the generation of association rules more than mere frequency (i.e., support) of the pattern. For example, one can set the minimum support (min_sup) at a certain value but get results of only frequent 1-sequences, which has no contribution to association rules and minimal effect on decision making, especially in applications where frequent 1-sequences are not really interesting. As a matter of fact, we have noticed during our comparative experiments with sequential pattern mining algorithms, that in most cases a min_sup more than 1% generates only frequent 1-sequences, if it does generate any. Geng and Hamilton [2006] have identified nine criteria that can be applied in three different ways to determine if a pattern is interesting or not (conciseness, coverage, reliability, peculiarity, diversity, novelty, surprisingness, utility, and actionability), of which we can list the size of the frequent pattern under utility and actionability, meaning that the pattern is of importance to the decision maker and it enables him/her to take further action. Aljandal [2009] discusses that itemset size is a propoerty which has not been involved directly in any interestingness measure, and he points out its importance in DNA replication applications. A direct application of frequent pattern size would be to use it as a limiting criteria during pattern mining and restrict the algorithm to find patterns of a certain minimum size. This can be applied in Pattern-Growth algorithms (like PrefixSpan) by allowing a minimum size of the prefix/suffix from which growth will start. Some challenges still remain, like what is the maximum size allowed given the provided min_sup value? What relation is there between min_sup and the size of found frequent patterns? How will this affect time and space complexity of an algorithm? The development of a mining method to answer these questions and also relate frequent pattern size to the utility of the pattern is left for future work.

8.2.4   Measuring User Interest by Page View Time

It can be claimed that the interest of a user in some item is related to the time he spends viewing the item's web page. This can be calculated from the web log since each entry is associated with a timestamp. It should be noted though, that if this viewing period

exceeds a certain limit it is considered that the user has ended the browsing session, and the next item belongs to another browsing session. This process of sessionizing has a lot of debate about it [Pabarskaite and Raudys, 2007; Huntington et al., 2008].

With sessionizing aside, if an importance value can be associated with an item based on the time spent by a user or all users viewing this item, then it can be used in ranking items for top-n recommendation, or for users clustering. In addition, such importance value can also be associated with support counting, and integrated into an interstingness measure for mining the web log. This way only frequent patterns of importance to the users are mined. Different levels of importance based on the time period can be defined by the data mining engineer, that help the decision maker assess association rules.

8.2.5   Semantic-rich Neural Networks

Since semantic information can be infused into a markov model, then a semantic-rich neural network is not far from being achieved. There are several models of neural networks, and there is no clear method for determining the best model to use, the best size of a model to use, and the optimal parameter for that model. All these factors, are challenging problems to the engineer when it comes to integrating neural networks with semantic information, most importantly in which part of the network to integrate this knowledge, in the sigmoid function, or simply to prune states from layers?

Other research directions and future work include studying the robustness of the top-n WRS to tag spamming, the use of domain ontology in CF WRS where users provide scores to items, studying the effect of ontology detail on the accuracy and performance of ontology-based WRS, and cross-ontology recommendation by aligning ontologies and gathering semantic information from multiple ontologies. Multidimensional information, like user demographics and gender can have a dramatic effect on the interpretation and utility of recommendation results.

Bibliography

Acharyya, S. and Ghosh, J. (2003). Context-sensitive modeling of web-surfing behaviour using concept trees. In *Proc. of the 5th WEBKDD Workshop.* 33

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734 – 749. 29

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, New York, NY, USA. ACM. 14, 19, 60, 72

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of 1994 International Conference on Very Large Databases (VLDB'94)*, pages 487–499. 14, 16

Agrawal, R. and Srikant, R. (March 1995). Mining sequential patterns. In *Proceedings of the 11th Int'l Conference on Data Engineering (ICDE-95)*, pages 3–14. 1, 14, 16, 19, 56, 58, 60

Aljandal, W. A. (2009). *Itemset size-sensitive interestingness measures for association rule mining and link prediction.* PhD thesis, Kansas State University, Manhattan, KS, USA. 98

Anand, S. S., Kearney, P., and Shapcott, M. (2007). Generating semantically enriched user profiles for web personalization. *ACM Trans. Internet Tech.*, 7(4). 30

Aussenac-gilles, N. (2005). Supervised text analysis for ontology and terminology engineering. In *Proceedings of the Dagstuhl Seminar on Machine Learning for the Semantic Web*, pages 35–46. 27

Ayres, J., Flannick, J., Gehrke, J., and Yiu, T. (2002). Sequential pattern mining using a bitmap representation. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 429–435, New York, NY, USA. ACM. 16, 18

Baumgarten, M., Büchner, A. G., Anand, S. S., Mulvenna, M. D., and Hughes, J. G. (2000). User-driven navigation pattern discovery from internet data. In *WEBKDD '99: Revised Papers from the International Workshop on Web Usage Analysis and User Profiling*, pages 74–91, London, UK. Springer-Verlag. 32, 36

Berendt, B. (2002). Using site semantics to analyze, visualize, and support navigation. *Data Min. Knowl. Discov.*, 6(1):37–59. 29, 32

Berendt, B. and Spiliopoulou, M. (2000). Analysis of navigation behaviour in web sites integrating multiple information systems. *The VLDB Journal*, 9(1):56–75. 32, 34, 36

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, 284(5):34–43. 25

Billsus, D., Brunk, C. A., Evans, C., Gladish, B., and Pazzani, M. (2002). Adaptive interfaces for ubiquitous web access. *Commun. ACM*, 45(5):34–38. 29

Bouza, A. (2010). Mo-the movie ontology (http://www.movieontology.org). [Online; 26. Jan. 2010]. 63, 90

Cantador, I., Bellogín, A., and Vallet, D. (2010). Content-based recommendation in social tagging systems. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 237–240, New York, NY, USA. ACM. 35

Caraballo, S. A. (2001). *Automatic construction of a hypernym-labeled noun hierarchy from text*. PhD thesis, Brown University, Providence, RI, USA. 27

Cha, S.-H. (2007). Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307. 53

Chiu, D.-Y., Wu, Y.-H., and Chen, A. L. P. (2004). An efficient algorithm for mining frequent sequences by a new strategy without support counting. In *ICDE*, pages 375–386. IEEE Computer Society. 18, 23

Cimiano, P., Hotho, A., and Staab, S. (2004). Comparing conceptual, divisive and agglomerative clustering for learning taxonomies from text. In *European Conference on Artificial Intelligence*, volume 16, page 435. 27

Ciravegna, F., Chapman, S., Dingli, A., and Wilks, Y. (2004). Learning to harvest information for the semantic web. In Bussler, C., Davies, J., Fensel, D., and Studer, R., editors, *The Semantic Web: Research and Applications*, volume 3053 of *Lecture Notes in Computer Science*, pages 312–326. Springer Berlin / Heidelberg. 27

Cooley, R., Mobasher, B., and Srivastava, J. (1999). Data preparation for mining world wide web browsing patterns. *Journal of Knowledge and Information Systems*, 1(1):5–32. 33

Cremonesi, P., Koren, Y., and Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *RecSys '10: Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46, New York, NY, USA. ACM. 90, 91

Cunningham, P., Bergmann, R., Schmitt, S., Traphöner, R., Breen, S., and Smyth, B. (2001). Websell: Intelligent sales assistants for the world wide web. *Künstliche Intelligenz*, 15(1):28–32. 31

Dai, H. and Mobasher, B. (2003). A road map to more effective web personalization: Integrating domain knowledge with web usage mining. In *Proceedings of the International Conference on Internet Computing (IC'03)*. 2, 30, 32, 33, 36

de Gemmis, M., Lops, P., Semeraro, G., and Basile, P. (2008). Integrating tags in a semantic content-based recommender. In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pages 163–170, New York, NY, USA. ACM. 34

Deshpande, M. and Karypis, G. (2004). Selective markov models for predicting web page accesses. *Transactions on Internet Technology*, 4(2):163–184. 36, 67, 71, 73, 74, 75, 77

Diederich, J. and Iofciu, T. (2006). Finding communities of practice from user profiles based on folksonomies. In Tomadaki, E. and Scott, P. J., editors, *EC-TEL Workshops*, volume 213 of *CEUR Workshop Proceedings*. CEUR-WS.org. 34

Doan, A., Domingos, P., and Levy, A. (2000). Learning source descriptions for data integration. In *presented at the International Workshop on The Web and Databases (WebDB)*, pages 81–86. 27

Domingue, J. B. (1998). Tadzebao and webonto: Discussing, browsing and editing ontologies on the web. In *Proceedings of the Knowledge Acquisition Workshop*. 45

Dong, G. and Pei, J. (2007). *Sequence Data Mining*. Springer, first edition. 18

Eilbeck, K., Lewis, S., Mungall, C., Yandell, M., Stein, L., Durbin, R., and Ashburner, M. (2005). The sequence ontology: a tool for the unification of genome annotations. *Genome biology*, 6(5):R44. 28

Eirinaki, M. and Vazirgiannis, M. (2003). Web mining for web personalization. *ACM Trans. Internet Technol.*, 3(1):1–27. 30

Eirinaki, M., Vazirgiannis, M., and Varlamis, I. (2003). Sewep: using site semantics and a taxonomy to enhance the web personalization process. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, New York, NY, USA. ACM. 29, 33, 35, 36, 81

El-Sayed, M., Ruiz, C., and Rundensteiner, E. A. (2004). Fs-miner: efficient and incremental mining of frequent sequence patterns in web logs. In *WIDM '04: Proceedings of the 6th annual ACM international workshop on Web information and data management*, pages 128–135, New York, NY, USA. ACM. 8

Ezeife, C. I. and Lu, Y. (2005). Mining web log sequential patterns with position coded pre-order linked wap-tree. *Data Mining and Knowledge Discovery*, 10(1):5–38. 5, 8, 10, 14, 17, 36, 61, 64

Facca, F. M. and Lanzi, P. L. (2003). Recent developments in web usage mining research. In Kambayashi, Y., Mohania, M. K., and Wöß, W., editors, *DaWaK*, volume 2737 of *Lecture Notes in Computer Science*, pages 140–150. Springer. 14

Facca, F. M. and Lanzi, P. L. (2005). Mining interesting knowledge from weblogs: a survey. *Data and Knowledge Engineering*, 53(3):225–241. 8, 14

Fenstermacher, K. D. and Ginsburg, M. (2002). Mining client-side activity for personalization. In *WECWIS '02: Proceedings of the Fourth IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS'02)*, page 205, Washington, DC, USA. IEEE Computer Society. 8

Geng, L. and Hamilton, H. J. (2006). Interestingness measures for data mining: A survey. *ACM Computing Survey*, 38. 98

Glover, E., Pennock, D. M., Lawrence, S., and Krovetz, R. (2002). Inferring hierarchical descriptions. In *Proceedings of the eleventh international conference on Information and knowledge management*, CIKM '02, pages 507–514, New York, NY, USA. ACM. 27

Goethals, B. (2005). Frequent set mining. In Maimon, O. and Rokach, L., editors, *The Data Mining and Knowledge Discovery Handbook*, pages 377–397. Springer. 8

Guan, Z., Wang, C., Bu, J., Chen, C., Yang, K., Cai, D., and He, X. (2010). Document recommendation in social tagging services. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 391–400, New York, NY, USA. ACM. 35

Guo, J., Keselj, V., and Gao, Q. (2005). Integrating web content clustering into web log association rule mining. In Kégl, B. and Lapalme, G., editors, *Canadian Conference on AI*, volume 3501 of *Lecture Notes in Computer Science*, pages 182–193. Springer. 32

Hahn, U. and Schnattinger, K. (1998). Ontology engineering via text understanding. In *Proceedings of the 15th Wolrd Computer Congress 'The Global Information Society on The Way to The Next Millenium' (IFIP'98*, pages 429–442. 27

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., and Hsu, M. (2000). Freespan: frequent pattern-projected sequential pattern mining. In *Proc. 2000 Int. Conf. Knowledge Discovery and Data Mining (KDD'00)*, pages 355–359. 17

Handschuh, S. and Staab, S. (2002). Authoring and annotation of web pages in cream. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 462–473, New York, NY, USA. ACM. 45

Hastings, P. M. (1994). *Automatic acquisition of word meaning from context*. PhD thesis, University of Michigan, Ann Arbor, MI, USA. 27

Hayes, C., Avesani, P., and Veeramachaneni, S. (2007). An analysis of the use of tags in a blog recommender system. In Veloso, M. M., editor, *IJCAI*, pages 2772–2777. 34

Hearst, M. (1998). Automated discovery of WordNet relations. *WordNet: an electronic lexical database*, pages 131–151. 27

Hofmann, T. (2003). Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266, New York, NY, USA. ACM. 30

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115. 30

Hu, B., Dasmahapatra, S., and Lewis, P. H. (2007). Semantic metrics. *Intl Journal of Metadata, Semantics and Ontologies*, 2(4):242–258. 25, 53

Huntington, P., Nicholas, D., and Jamali, H. R. (2008). Website usage metrics: A re-assessment of session data. *Information Processing and Management: an International Journal*, 44:358–372. 10, 99

Inkpen, D. Z. and Hirst, G. (2003). Automatic sense disambiguation of the near-synonyms in a dictionary entry. In *Proceedings of the 4th international conference on Computational linguistics and intelligent text processing*, CICLing'03, pages 258–267, Berlin, Heidelberg. Springer-Verlag. 27

Iváncsy, R. and Vajk, I. (2006). Frequent pattern mining in web log data. *Acta Polytechnica Hungarica, Journal of Applied Science at Budapest Tech Hungary, Special Issue on Computational Intelligence*, 3(1):77–90. 8, 14

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579. 4, 51

Jin, X., Zhou, Y., and Mobasher, B. (2004). Web usage mining based on probabilistic latent semantic analysis. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 197–205, New York, NY, USA. ACM. 36

Jin, X., Zhou, Y., and Mobasher, B. (2005). A maximum entropy web recommendation system: combining collaborative and content features. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 612–617, New York, NY, USA. ACM. 30

Joachims, T., Freitag, D., and Mitchell, T. M. (1997). Web watcher: A tour guide for the world wide web. In *IJCAI (1)*, pages 770–777. 29

Li, J. and Zaïane, O. (2004). Combining usage, content, and structure data to improve web site recommendation. *E-Commerce and Web Technologies*, LNCS 3182:305–315. 30, 32, 38

Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80. 31

Lu, H., Luo, Q., and Shun, Y. K. (2003). Extending a web browser with client-side mining. In Zhou, X., Zhang, Y., and Orlowska, M. E., editors, *APWeb*, volume 2642 of *Lecture Notes in Computer Science*, pages 166–177. Springer. 8

Luke, S., Spector, L., Rager, D., and Hendler, J. (1997). Ontology-based web agents. In *Proceedings of the 1st Intl. Conf. on Autonomous Agents*, pages 59–66. 45

Mabroukeh, N. R. (1998). Training artificial neural networks to pronounce arabic text. Master's thesis, University of Jordan, Amman, Jordan. 77

Mabroukeh, N. R. and Ezeife, C. I. (2009a). Semantic-rich markov models for web prefetching. In *IEEE International Conference on Data Mining Workshops*, pages 465–470. 3, 65

Mabroukeh, N. R. and Ezeife, C. I. (2009b). Using domain ontology for semantic web usage mining and next page prediction. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 1677–1680, New York, NY, USA. ACM. 3, 55, 65, 78

Mabroukeh, N. R. and Ezeife, C. I. (2010). A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys*, 43:3:1–3:41. 1, 3, 6, 10, 14, 20, 22

Mabroukeh, N. R. and Ezeife, C. I. (2011). Ontology-based web recommendation from tags. In *IEEE International Conference on Data Engineering Workshops (ICDEW)*. 3, 78

Maedche, A., Pekar, V., and Staab, S. (2002). Ontology learning part one - on discovering taxonomic relations from the web. In *Proceedings of the Web Intelligence conference*, pages 301–322. Springer Verlag. 27

Manning, C. D. and Schuetze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press. 27

Melville, P., Mooney, R. J., and Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In *AAAI/IAAI*, pages 187–192. 30

Meo, R., Lanzi, P. L., Matera, M., and Esposito, R. (2004). Integrating web conceptual modeling and web usage mining. In Mobasher, B., Nasraoui, O., Liu, B., and Masand, B. M., editors, *WebKDD*, volume 3932 of *Lecture Notes in Computer Science*, pages 135–148. Springer. 9

Middleton, S. E., Roure, D. D., and Shadbolt, N. R. (2009). Ontology-based recommender systems. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, International Handbooks Information System, pages 779–796. Springer Berlin Heidelberg. 33

Middleton, S. E., Shadbolt, N., and Roure, D. D. (2004). Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.*, 22(1):54–88. 33

Miller, G., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. (1990). Introduction to wordnet: An on-line lexical database. *International Journal of lexicography*, 3(4):235. 84

Missikoff, M., Navigli, R., and Velardi, P. (2002). Integrated approach to web ontology learning and engineering. *IEEE Computer*, 35(11):60–63. 27

Mobasher, B. (2006). Web usage mining. In Liu, B., editor, *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data*, pages 449–483. Springer-Verlag, Berlin, Heidelberg. 9

Mobasher, B., Cooley, R., and Srivastava, J. (2000). Automatic personalization based on web usage mining. *Commun. ACM*, 43(8):142–151. 31

Mobasher, B., Dai, H., Luo, T., and Nakagawa, M. (2002). Discovery and evaluation of aggregate usage profiles for web personalization. *Data Min. Knowl. Discov.*, 6(1):61–82. 30

Mobasher, B., Jin, X., and Zhou, Y. (2004). Semantically enhanced collaborative filtering on the web. *Web Mining: From Web to Semantic Web*, LNAI 3209:57–76. 30

Niwa, S., Doi, T., and Honiden, S. (2006). Web page recommender system based on folksonomy mining for itng '06 submissions. In *ITNG*, pages 388–393. IEEE Computer Society. 34

Oberle, D., Berendt, B., Hotho, A., and Gonzalez, J. (2003). Conceptual user tracking. In Ruiz, E. M., Segovia, J., and Szczepaniak, P. S., editors, *AWIC*, volume 2663 of *Lecture Notes in Computer Science*, pages 155–164. Springer. 34, 35

Pabarskaite, Z. and Raudys, A. (2007). A process of knowledge discovery from web log data: Systematization and critical review. *Journal of Intelligent Information Systems*, 28:79–114. 10, 32, 99

Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. (2001). Prefixspan: Mining sequential patterns by prefix-projected growth. In *Proceedings of the 17th International Conference on Data Engineering*, pages 215–224. IEEE Computer Society. 13, 14, 17, 19, 36

Pei, J., Han, J., Mortazavi-Asl, B., and Zhu, H. (2000). Mining access patterns efficiently from web logs. In *PADKK '00: Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, pages 396–407, London, UK. Springer-Verlag. 8, 14, 17

Pereira, F., Tishby, N., and Lee, L. (1993). Distributional clustering of english words. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, ACL '93, pages 183–190, Stroudsburg, PA, USA. Association for Computational Linguistics. 27

Pitkow, J. and Pirolli, P. (1999). Mining longest repeating subsequences to predict www surfing. In *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems 2*, pages 13–21. 8, 71, 77

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137. 85

Rydin, S. (2002). Building a hyponymy lexicon with hierarchical structure. In *Proceedings of the ACL-02 workshop on Unsupervised lexical acquisition - Volume 9*, ULA '02, pages 26–33, Stroudsburg, PA, USA. Association for Computational Linguistics. 27

Rymon, R. (1992). Search through systematic set enumeration. In *Proc. of Third Int'1 Conf. on Principles of Knowledge Representation and Reasoning*, pages 539–550. 11

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523. 33

Sanderson, M. and Croft, B. (1999). Deriving concept hierarchies from text. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 206–213, New York, NY, USA. ACM. 27

Sandvig, J. J., Mobasher, B., and Burke, R. (2007). Robustness of collaborative recommendation based on association rule mining. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 105–112, New York, NY, USA. ACM. 30

Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295. 30

Schwab, I., Kobsa, A., and Koychev, I. (2001). Learning user interests through positive examples using content analysis and collaborative filtering. *Internal Memo, GMD*. 31

Simpson, T. and Dao, T. (2010). Wordnet-based semantic similarity measurement. `http://www.codeproject.com/KB/string/semanticsimilaritywordnet.aspx`. 85

Song, S., Hu, H., and Jin, S. (2005). Hvsm: A new sequential pattern mining algorithm using bitmap representation. In Li, X., Wang, S., and Dong, Z. Y., editors, *ADMA*, volume 3584 of *Lecture Notes in Computer Science*, pages 455–463. Springer. 16, 18

Spiliopoulou, M. (2000). Web usage mining for web site evaluation. *Commun. ACM*, 43(8):127–134. 29

Srikant, R. and Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the 5th Int'l Conference on Extending Database Technology: Advances in Database Technology*, pages 3–17. 56, 59

Srivastava, J., Cooley, R., Deshpande, M., and Tan, P.-N. (2000). Web usage mining: discovery and applications of usage patterns from web data. *ACM SIGKDD Explorations Newsletter*, 1(2):12–23. 8, 14

Stumme, G., Hotho, A., and Berendt, B. (2006). Semantic web mining: State of the art and future directions. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 4(2):124–143. 24

Teisseire, F. M. M. and Poncelet, P. (2005). Sequential pattern mining: A survey on issues and approaches. In *in Encyclopedia of Data Warehousing and Mining, Information Science Publishing*, pages 3–29. Oxford University Press. 14

Vanzin, M. and Becker, K. (2006). Ontology-based rummaging mechanisms for the interpretation of web usage patterns. In Ackermann, M., Berendt, B., Grobelnik, M., Hotho, A., Mladenic, D., Semeraro, G., Spiliopoulou, M., Stumme, G., Svtek, V., and van Someren, M., editors, *Semantics, Web and Mining*, volume 4289 of *Lecture Notes in Computer Science*, pages 180–195. Springer Berlin / Heidelberg. 81

Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A., and Ciravegna, F. (2002). Mnm: Ontology driven semi-automatic and automatic support for semantic

markup. In *Proceedings of the 13th Intl. Conf. on Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pages 379–391. 45

W3C (1999). Web characterization terminology & definitions sheet. 10

Wang, J. and Han, J. (2004). Bide: Efficient mining of frequent closed sequences. In *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*, page 79, Washington, DC, USA. IEEE Computer Society. 8, 13, 23

Witten, I. and Frank, E. (1999). *Data Mining: Practical machine learning tools and techniques with java implementations.* Morgan Kaufmann Pub. 34

Wu, Z. and Palmer, M. S. (1994). Verb semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics. 51, 85

Xu, F. (2002). Term extraction and mining of term relations from unrestricted texts in the financial domain. In *Proceedings of BIS.* 27

Yang, Z. and Kitsuregawa, M. (2005). Lapin-spam: An improved algorithm for mining sequential pattern. In *ICDEW '05: Proceedings of the 21st International Conference on Data Engineering Workshops*, page 1222, Washington, DC, USA. IEEE Computer Society. 16, 18

Yang, Z., Wang, Y., and Kitsuregawa, M. (2005). Lapin: effective sequential pattern mining algorithms by last position induction. Technical report, Info. and Comm. Eng. Dept., Tokyo University. http://www.tkl.iis.u-tokyo.ac.jp/ yangzl/Document/LAPIN.pdf. 13, 18, 23

Yang, Z., Wang, Y., and Kitsuregawa, M. (2006). An effective system for mining web log. In Zhou, X., Li, J., Shen, H. T., Kitsuregawa, M., and Zhang, Y., editors, *APWeb*, volume 3841 of *Lecture Notes in Computer Science*, pages 40–52. Springer. 13, 18

Yang, Z., Wang, Y., and Kitsuregawa, M. (2007). Lapin: Effective sequential pattern mining algorithms by last position induction for dense databases. In Ramamohanarao, K., Krishna, P. R., Mohania, M. K., and Nantajeewarawat, E., editors, *DASFAA*, volume 4443 of *Lecture Notes in Computer Science*, pages 1020–1023. Springer. 18

Zaki, M. J. (1998). Efficient enumeration of frequent sequences. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 68–75, New York, NY, USA. ACM. 14, 16

Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1-2):31–60. 17

Zanardi, V. and Capra, L. (2008). Social ranking: uncovering relevant content using tag-based recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pages 51–58, New York, NY, USA. ACM. 34

Zhang, Y., Callan, J., and Minka, T. (2002). Novelty and redundancy detection in adaptive filtering. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 81–88, New York, NY, USA. ACM. 29

Zhou, L. (2007). Ontology learning: state of the art and open issues. *Information Technology and Management*, 8:241–252. 27, 28

Ziegler, C.-N., McNee, S. M., Konstan, J. A., and Lausen, G. (2005). Improving recommendation lists through topic diversification. In Ellis, A. and Hagino, T., editors, *WWW*, pages 22–32. ACM. 33, 34

Vita Auctoris

Nizar Mabroukeh was born in 1974 in Riyadh of Saudi Arabia to Radi and Wedad Mabroukeh from Palestine. Nizar finished his high school and graduated from Al-Hussein College in Amman-Jordan in 1991. He went on to study Computer Science at The University of Jordan and received his Bachelor of Science degree in 1995 and his Master's of Science degree in Computer Science as well in 1998. Nizar has filled several industrial and academic positions until the year 2007 in which he pursued doctoral studies and recevied his PhD degree in Computer Science from the University of Windsor in Ontario-Canada in Winter 2011.