WEB USAGE MINING AND RECOMMENDATION WITH SEMANTIC INFORMATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SÜLEYMAN SALIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

FEBRUARY 2009

Approval of the thesis:

WEB USAGE MINING AND RECOMMENDATION WITH SEMANTIC INFORMATION

submitted by **SÜLEYMAN SALIN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** ________________

Prof. Dr. Müslim Bozyiğit
Head of Department, **Computer Engineering** ________________

Assist. Prof. Dr. Pınar Şenkul
Supervisor, **Computer Engineering Department, METU** ________________

Examining Committee Members:

Prof.Dr. İsmail Hakkı Toroslu
Computer Engineering, METU ________________

Assist. Prof. Dr. Pınar Şenkul
Computer Engineering, METU ________________

Assoc. Prof. Ferda Nur Alpaslan
Computer Engineering, METU ________________

Assist. Prof. Dr. Tolga Can
Computer Engineering, METU ________________

Dr. Ersin ELBAŞI
Computer Engineering, TÜBİTAK ________________

Date: ________________

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: SÜLEYMAN SALIN

Signature :

ABSTRACT

WEB USAGE MINING AND RECOMMENDATION WITH SEMANTIC INFORMATION

Salın, Süleyman
M.S., Department of Computer Engineering
Supervisor : Assist. Prof. Dr. Pınar Şenkul

February 2009, 133 pages

Web usage mining has become popular in various business areas related with Web site development. In Web usage mining, the commonly visited navigational paths are extracted in terms of Web page addresses from the Web server visit logs, and the patterns are used in various applications. The semantic information of the Web page contents is generally not included in Web usage mining. In this thesis, a framework for integrating semantic information with Web usage mining is implemented. The frequent navigational patterns are extracted in the forms of ontology instances instead of Web page addresses and the result is used for making page recommendations to the visitor. Moreover, an evaluation mechanism is implemented to find the success of the recommendation. Test results proved that stronger and more accurate recommendations are obtained by including semantic information in the Web usage mining instead of using on visited Web page addresses.

Keywords: Semantic Information, Web Usage Mining, SPADE, Web Page Recommendation

ÖZ

ANLAMSAL BİLGİLER İLE WEB KULLANIM MADENCİLİĞİ VE ÖNERİM

Salın, Süleyman
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Yar. Doç. Dr. Pınar Şenkul

Şubat 2009, 133 sayfa

Web kullanım madenciliği Web site geliştirme ile ilgili iş alanlarında oldukça popüler olmuştur. Web kullanım madenciliği, Web sunucularının tuttuğu ziyaret kayıtlarından, ziyaretçilerin sıkça yaptığı sayfa gezme örneklerini Web sayfası adresi üzerinden bulma işlemidir ve bulunan bu desenler bir çok alanda kullanılmaktadır. Web içeriklerinin anlamsal bilgileri genellikle Web kullanım madenciliğinde kullanılmamaktadır. Bu tezde, anlamsal bilgileri Web kullanım madenciliğine eklemek için bir çerçeve geliştirilmiştir. Sıkça yapılan sayfa gezme örnekleri, Web sayfa adresi biçimi yerine, anlamsal bilgi örnekleri biçiminde bulunacaktır ve bulunan sonuçlar ziyaretçiye sayfa önerisi yapmak için kullanılacaktır. Bunun yanında yapılan önerilerin başarısını bulmak için bir değerlendirme tekniği geliştirilmiştir. Test sonuçları Web kullanım madenciliğinde Web sayfaları yerine, anlamsal bilgiler kullanıldığında daha kuvvetli ve doğru öneriler yapıldığını göstermiştir.

Anahtar Kelimeler: Anlamsal Bilgiler, Web Kullanım Madenciliği, SPADE, Web Sayfaları Önerisi

To my family...

ACKNOWLEDGMENTS

I would like to present my special thanks to my supervisor Assist. Prof. Dr. Pınar Şenkul for her guidance, understanding and encouragement throughout the development of this thesis.

I also extend my thanks to my employer for providing me with time to study whenever I needed it.

I would like to give my appreciation to Seçkin Yayıncılık San. Tic. A.Ş. and also to the Department of Computer Engineering of METU for supporting me in finishing my thesis by providing the necessary datasets for the evaluation of the implemented system.

I thank all of my friends for their patience and unforgettable help. Finally, words cannot truly express my deepest gratitude to my family, who always give me their love, emotional support, motivation, and the encouragement to finish my study.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

The Semantic Web is one of the most popular subjects of the recent era. The main aim of the Semantic Web is to make Web content understandable not only by humans, but also computers, thus permiting the software agents to search for and find desired content, share information, and share knowledge in a format that other software agents can understand. In Tim Berners-Lee's own words "The Semantic Web is not a separate Web, but an extension of the current one, in which information is given well-defined meaning, better enabling computers, and people to work in cooperation."

Currently, Web pages are designed to be read by humans. People can read books, get information about movies, and buy books or music CDs on the Internet. However, software agents like browsers do not have information about the content. Therefore, they cannot understand the difference between a news Web site and one about shopping. It needs human directions for a software agent to do the desired task. With the realization of Semantic Web, software agents can easily "find the cheapest *Harry Potter* book" or "find all the Web sites that sell 2mp Sony Web cams."

Recommendation is very important for Web site owners, especially when it is an online shopping site. Web site developers can recommend frequently visited pages to the visitor to keep them on the site longer, and make them buy more items from the Web site. Generally, web pages contain one or more dialogs that recommend several web pages to the user, as shown in Figure 1.1. Recommendations are made by joining collaborative navigation patterns with the user current navigation. Therefore, finding collaborative navigation patterns is the first step of recommendation. To find such common navigation patterns, Web usage mining was introduced as a branch of data mining.

Figure 1.1: Recommendation samples from www.amazon.com

Web usage mining is extracting the frequent navigation paths by analyzing Web server log files. The results are used mainly to recommend to the visitor what they might like to visit at the next step, or recommend a product by looking at what they bought earlier. Besides this, the result of Web usage mining can be helpful to the Web site owner in making decisions, such as which content group of the Web site should be enriched, or, if it is a shopping Web site, which products should be promoted, or whether to replace the contents of the most visited web pages. Generally, the Web site visit log file is extremely huge, so current studies emphasize making the data mining algorithm faster and more accurate.

One of the most important disadvantages of the current approaches in Web usage mining is that the result is produced in terms of Web pages (i.e. web page addresses); hence, there is no semantic meaning of the common navigation profile. Site owners generally want to learn which category or group is preferred, rather than which page is more visited. To obtain such results, the resulting pattern of the Web usage mining must be analyzed by humans, and this process can be erroneous and time-consuming.

Another disadvantages of classical Web usage mining is called the new-item problem which entails failing to recommend a newly added page or product to the visitors since this product or page is not in the current common navigation profiles. To overcome this, the common navigation profile can be extracted in terms of semantic information so that the common navigation profile will be in ontological terms and concepts. Therefore, newly added items can be recommended to the user as long as this item's concept is in the common navigation profiles.

The aim of this thesis is to cover the shortcomings of Web usage mining by analyzing the visit log, not only by Web page addresses but also by the classes and properties that come together with semantic information of the Web site. The thesis brings together these two popular subjects, Semantic Web and Web Usage Mining. By scanning a Web site visit log, the common usage profile in terms of ontological classes and individuals will be found. Following that, these results will be used in recommending subsequent pages to the user.

To date, the number of studies on Web usage mining with semantic information is very low. Mostly, researchers emphasize extracting semantic ontologies by applying content mining algorithms[4, 30]. There are studies that aim to generate patterns in terms of semantic information as in the proposed studies [4,5, 6]. Generally, these works map out the results of classical Web usage mining with ontological terms and concepts. In addition, some of these studies use clustering techniques to recommend a new page to the visitor. In the proposed work, unlike previous approaches, sequential association rule mining algorithms will be used, rather than clustering algorithms. Through sequential rule mining, the results will be in a sequential form that permits us to know which pages are visited and in what order.

In this study, a different approach is employed from the previous approaches. Generally one class in ontology has a substantial number of attributes(properties). For example, a movie class has casting, director, and target genre and so on. Some people navigates in the Web site, only considering the casting attribute and some other people may consider only director attribute of the movie class. Since navigation paths of the visitors are strongly affected by the attributes of an class, in this thesis, a different approach, which seperately finds the frequent navigation paths in terms of all distinct attributes of a particular ontology class. The frequent navigation paths, which are generated by some web usage mining algorithms, are used to generate recommendation for the user. Recommendations either are generated considering the frequent navigation paths in terms of a single class or considering combinition of frequent navigation paths in terms of a several classes. The evaluations of the both recommendations are presented in the evaluation section of the thesis.

The proposed system generates more meaningful frequent navigation patterns for web site owners. Including semantic information allows web site owners to understand the purpose of the visitors when they are navigating around the web site. This allows the web site owners to make more accurate decisions about promotions, or web content placement.

As the first step of the thesis, sequential pattern mining algorithms are evaluated and the selected one is implemented under required modifications. The algorithm reads the Web site log and matches the entries to the ontological classes and individuals, and finds the sequential common usage profile of visitors in terms of ontological classes. According to the results, pages containing ontological entities that the user will probably visit according to common visit patterns are recommended to the user.

The proposed approach is applied to the visit logs of three different Web sites, two of which are visit logs of an online and real Web site. The results are evaluated by comparing the recommended page with the visitor's actual next page. The success of the proposed system is measured by an evaluation mechanism. The datasets are divided into training and evaluation, and the recommendations which are generated from the training dataset are tested on the evaluation dataset. Many different evaluations are performed on the datasets. The results show that including semantic information produces more recommendations to the user, and increases the accuracy of those recommendations. The users get more exact recommendations. A detailed discussion of the evaluations is provided in Chapter 4.

1.2 Organization of the thesis

Chapter 2 provides a brief overview of the Semantic Web and the related research on Web usage mining, Web usage mining on the Semantic Web, and a recommendation system. A brief history of the Semantic Web will be mentioned. The two sequential data mining algorithms, SPADE and GSP, which are used in the thesis are introduced. After that, some notable research about Web usage mining on the Semantic Web will be mentioned. Finally, a brief introduction of the recommendation systems will be given.

Chapter 3 provides a detailed discussion of the proposed system's design and implementation issues. The topics of the system design are visit log parsing, common pattern finding in terms of ontological terms, recommendation and evaluation of results. Moreover implementation details and implemented system features will be mentioned at the end of this chapter.

Chapter 4 covers test results of the implementation. The implementation is tested on some data sets and the result of the test will be discussed in detail in this chapter.

Chapter 5 provides the conclusion and possible future work directions for the Web usage mining on semantic Webs.

CHAPTER 2

LITERATURE SURVEY AND BASIC CONCEPTS

2.1 Semantic Web

The Semantic Web is an extenstion to the World Wide Web which makes it possible to understand, share and use the content of the web. The following sections describe the past, current and future studies of the Semantic Web.

2.1.1 History

The Semantic Web is based on the vision and ideas of Sir Tim Berners-Lee, who is the inventor of the WWW. According to his vision, huge amounts of data on the internet are only understandable and interpretable by humans; machine support is very limited. In order to support the user in his task, the Web should be enriched by the machines' ability to process the information. Therefore, the Web content and objects should be semantically introduced to the machine world.

The Semantic Web is in the development stage, so not all features have yet been fully standardized. Some are still in the development phase by W3C [34] working groups. So far, the Semantic Web working groups proposed RDF in 2002[35] and OWL in 2004[36], thus, the ontology of the Web object in terms of classes, properties of classes, and individuals can be defined in ontology files.

2.1.2 Overview of the current Situation

The RDF and Web Ontology Language (OWL) permit us to define domain ontology to model the semantics of the Web objects. Domain ontology contains all the classes in the domain space, and the relations among them. Ontology definition starts with extracting the concepts and terms from the domain space, and then defining all the known relationships among them. In addition, ontology individuals should be defined, and mapping between Web objects and ontology individuals of the domain should be done. Therefore, a software agent can obtain the desired information by looking at the mapping between domain ontology and the Web content.

2.1.3 Future works and projects

There are limitations on the Semantic Web. Reluctance major companies, lack of a killer application and certain disadvantages of ontology definition (i.e. lack of globally accepted ontologies, difficulty in defining ontology by hand) restrict usage of the Semantic Web worldwide. However, it is commonly envisioned in the recent future that the Semantic Web will be accepted globally. According to TopQuadrant [37], which is a consulting firm that specializes in Semantic Web technologies, the market for semantic technologies will have an annual growth rate of between 60% and 70% until 2010.

In addition, the incomplete parts of the Semantic Web, especially those concerning trust and proof layers are going to be defined to check the validity of statements made in the Semantic Web. Very little is written about these layers, but they will be interesting topics for future researchers.

2.1.4 Overview

2.2 Web Usage Mining

Data mining in other words "knowledge discovery" emerged as a result of the need to discover interesting rules from a large collections of data. Thus, data mining is the process of extracting invisible patterns from a large set of data. The aim is to transform the huge jumble of data into knowledge.

Association rule mining, a branch of data mining, is the process of extracting the relationship between items and was introduced in [1]. Association rule mining is finding the expression X $\Rightarrow$ Y, where X and Y are a set of items, from a given set of transaction databases. The meaning of such a rule is that transactions in the database which contains the items in X tend to contain the items in Y. If the given transaction database keeps the shopping transactions of a marketplace, then such rule means that customers who buy X tend to buy Y. If the given transaction database is a Web site visit log, then such a rule means visitors who visit page X tend to visit Y.

The support of rule X $\Rightarrow$ Y is the percentage of the transactions that contain both X and Y to all transactions. The confidence is the percentage of the transaction that contains Y to the transaction that contains X. From a given database, association rule mining is the process of finding all the rules that have support and confidence greater than given minimum support and minimum confidence respectively. There are several algorithms proposed for obtaining association rules [4, 1, 2, 3, 7, 25, 26].

Association rule mining is employed in many different application areas, including Web usage mining, intrusion detection, bioinformatics and market basket analysis. Web usage mining employs rule mining algorithms such as [1, 2, 3, 4, 7] to find visit relationships between Web-pages.

In Web usage mining, the aim is to find the frequent navigation paths of a web site by using Web server log files. The transaction database, which is the visit log of a Web site, contains the address of the Web objects, some information about the visitor such as the IP address, browser information, and the time of the visit of each request to any object of the Web site. To apply Web usage mining, the Web site log file should be converted to session-based transaction logs.

A session in the web servers (HTTP sessions) is established at a certain point in time for a certain client, and then torn down at a later time. The session is used for the purpose of interactive information exchange between the client and the web server. A user would make many web object requests from the web server in his session. In a session-based log, all the Web objects the user has visited in that session are written sequentially in order of access time. By applying some of the association rule mining algorithms the common usage profile can be extracted. For example, a Web site that sells computers and accessories can establish that people who visit a lazer printer page also visit a lazer printer toner page, and people who visit a computer page also visit the operating system page. Hence, Web site owners can recommend the page operation system and lazer printer toner to the users who are visiting the computer or lazer printer pages respectively. In Web usage mining, it is possible to find visit patterns, without order information, as shopping transactions or the visit order would be considered, and sequential patterns and rules are generated.

2.2.1 Sequential Association Rule Mining Algorithms

Association rule mining is used to discover interesting relations among items in a large database. An example from market basket analysis would be that people who buy potatoes would also buy beer. Sequential association rule mining is a limited version of association rule mining. In sequential association rule mining, items in the both input (the dataset) and output (association rules) are presented in sequential form (mostly ordered by a timestamp). An example from web page recommendation would be that computer engineering students who visit the main page of the department, also later visit the web page of the courses they enrolled.

In this thesis, we aimed to obtain sequential patterns and rules. There are many sequential association rule mining algorithms. From these studies the following two sequential association rule mining algorithms are studied in the thesis.

2.2.1.1 Mining Generalized Association Rules

Srikant and Agrawal have proposed the sequential association rule mining algorithm ”Mining Generalized Association Rules” in [4]. The main difference between this algorithm and previous sequential association rule mining algorithms [1, 2] is considering the presence of taxonomy. Generalized association rule mining is different from ordinary association rule mining since some rules may hold true in generalized association rule mining but may not do so in association rule mining without taxonomy. An example from [4] will make the above statement clearer. From the given taxonomy, jacket is-an outwear and outwear is-a clothing. The rule ”people who buy outwear tend to buy shoes” can be inferred from the rule ”people who buy jacket tend to buy shoes” since jacket is-an outwear.

Three algorithms have been proposed in [4] order to find generalized association rules. The Basic algorithm is simple, but not very fast. Cumulative and Estimate techiques are faster but more complex. The main principle behind the basic algorithm is extending the transactions by adding all ancestors of the items. A new extended transaction is added to the database for each permutation of the transaction by replacing the item with its ancestor. Generalized association rules can be found by running any association rule mining algorithms [1][2] on the extended transaction database. The author used Appriori algorithm [2] to find association rules.

Algorithm 2.2.1: GSP()

F_1 = { frequent 1-sequences}
for $(k = 2; F_{k-1} \neq \emptyset; k = k + 1)$ **do**
  C_k = Set of candidate f-sequences;
  for all input-sequences ε in the database **do**
    increment count of all $\alpha \in C_k$ contained in ε
  $F_k = \{\alpha \in \mathrm{C}_k | a.sup \geq min\_sup\}$
Set of all frequent sequences $= \sqcup_k F_k$

The cumulative algorithm is the modification of the basic algorithm by adding several optimization techniques. The first optimization is filtering the ancestors that added to the transactions in the past. At each pass, the number of transactions can be reduced by not replacing the items with any items' ancestors that are non-frequent. The second optimization is the pre-computing of the ancestors. By creating a data structure to keep the taxonomy, it is not required to compute the taxonomy at each pass. In addition, dropping non-frequent ancestors from the data structure at each pass improves the performance. The last optimization is about pruning transactions containing an item and its ancestors. Since the support of a transaction that contains both the item and its ancestor is the same as the support of the transaction containing the item without its ancestors, so the ancestor is dropped.

One more optimization is added in [4] to the previous technique and the resulting technique is called "Estimate." Consider the taxonomy in the previous example, jacket is-an outwear and outwear is-a clothing. After the first phase, consider {Clothing}, {Shoes}, {Outwear}, {Jacket} are the frequent one-item itemsets. At the next step, {Clothing, Shoes} {Outwear, Shoes} {Jacket, Shoes} will be generated as candidate frequent itemsets. While counting the support of the candidates, if it is revealed that {Outwear, Shoes} is not a frequent itemset, then it is not necessary to compute the support count of {Jacket, Shoes} since Jacket is-an Outwear, hence, the support of it will be lower. Conversely, if {Outwear, Shoes} is a frequent itemset then {Clothes, Shoes} is definitely a frequent itemset since Outwear is a Clothes, but {Jacket, Shoes} may or may not be a frequent-itemset since support count of {Jacket, Shoes} is less than {Outwear, Shoes}.

Therefore, it is vital to know which candidate to count first, since correct guessing will eliminate several rules. In order to guess where to start, the support values of the candidates are calculated on a sample dataset. Then, the candidates that are expected to be frequent and whose items are taxonomically at the lowest level, and the candidates that are expected to be frequent and whose items are taxonomically at the upper levels are counted on real dataset. If those that are expected to be infrequent turn out to be infrequent, then the descendant candidates are eliminated, and if those that are expected to be frequent turn out to be frequent, then the ancestral candidates are eliminated. If the support counting contradicts the expectation, then the support of descendants or ancestors of the candidate will be calculated.

The authors evaluated the algorithms on two datasets, one is syntheticaly generated data set, the other one a real data set of a supermarket. The performance measure shows that Estimate and Cumulative algorithms are generally faster, by around 3 to 4 times, than the Basic. Estimate is 25% to 30% faster than Cumulative on many synthetic data set variations. The gap between Estimate and Cumulative increases when the number of transactions increases. Estimate and Cumulative shows a linear scale.

2.2.1.2 SPADE: An Efficient Algorithm for Mining Frequent Sequences

The main aim of the SPADE algorithm [7] is to overcome the burden of repeated database scans while calculating the support of the candidates. The problem is divided into sub-problems, and each sub-problem is calculated in the main memory using lattice techniques. The algorithm requires three database scans and does the job in memory by decomposing problems into sub-problems.

SPADE not only minimizes I/O costs by reducing database scans, but also minimizes computational costs by using efficient search schemes. The vertical ID-list based approach is also insensitive to data-skew.

The basic terminology used in SPADE is as follows. *Event* is a set of one or more items. The items in the event are unordered. *Sequence* is an ordered set of events. A sequence with k item is called k-sequence. For a sequence, event order is denoted as $<$ such that $\alpha_i < \alpha_j$ means the event α_i occurs before event α_j. $\leq$ denote subsequence relation. If $\alpha \leq \beta$ then there is one-to-one order preserving function f that maps events in α to events in β. Items of event β covers all items in the event α_j and sequence β is said to contain sequence α.

The support of a sequence α is the total number of sequences in the transaction database that contain α. A sequence is called a frequent sequence if the support is greater than a predefined minimum support count.

Table 2.1 from [7] shows vertical ID-list database format, where each sequence contains a list of event along with a timestamp, and each event has a set of items.

Table 2.1: Input-Sequence Database

SID	Time(EID)	Items
1	10	C D
1	15	A B C
1	20	A B F
1	25	A C D F
2	15	A B F
2	20	E
3	10	A B F
4	10	D G H
4	20	B F
4	25	A G H

Algorithm 2.2.2: SPADE($min\_sup, \mathcal{D}$)

F_1 = { frequent 1-sequences};

F_2 = { frequent 2-sequences};

ε = { equivalence classes $[\![X]\!]_{\theta 1}$};

for $all[\![X]\!] \in \varepsilon$ *do* Enumerate-Frequent-Seq($[\![X]\!]$);

Table 2.2 shows the association rules after running the algorithm on database depicted at Table 2.1. As shown in the frequent sequences, an event can be composed of one or more items such as the event BF in (D→BF→A) as long as the sequence is a subsequence of a sequence in the database. There can be arbitrary gaps among events of the frequent sequence as well.

After finding frequent sequences, the next step is identifying rules from the frequent sequences. From the frequent sequence of the database, sequence (D→B) is found in 2 sequences and (D→B→A) found in 2 sequences. Therefore, the probability of item A coming after the sequence (D→B) is 100%. Thus, the confidence of the rule (D→B) ⇒ (D→B→A) is 100%.

Table 2.2: Frequent Sequences of the database in the table [2.1]

Frequent 1-Sequences

A	4
B	4
D	2
F	4

Frequent 2-Sequences

AB	3
AF	3
B→A	2
BF	4
D→A	2
D→B	2
D→F	2
F→A	2

Frequent 3-Sequences

ABF	3
BF→A	2
D→BF	2
D→B→A	2
D→F→A	2

Frequent 4-Sequences

D→BF→A	2

SPADE is divided into three passes. In the first pass, the frequent 1-sequences are found. In the second pass, frequent 2-sequences are found. In the last pass, the transaction dataset is decomposed into prefix based equivalence classes, and each class is enumerated via Breadth-First-Search or Depth-First-Search in order to find 3-or-more-sequences.

In contrast to previous sequence association rule mining algorithms that use a horizontal database layout, SPADE uses a vertical layout database as shown in Table 2.1. In the vertical layout database, the ID-list for each item is maintained, instead of maintaining a set of sequences where each sequence has a set of events and each event has a set of items.

Given the vertical ID-list database, frequent 1-sequences are computed at one database scan by simply counting the distinct number of session ids for each item's ID-list.

Finding frequent 2-sequences is slightly more complex, but there are two ways to find frequent 2-sequences. The first one is ID-list joining of frequent 1-sequences with itself and counting the support of the result. The second way is to convert the vertical database to a horizontal database and compute all of the frequent 2-sequences by simply scanning over the sequences.

After finding frequent 1-sequence and 2-sequences, a class of each item is constructed in the memory as a sub-lattice S. Then, each class is enumerated in the memory; if the memory is not sufficient for the class, then several sub-lattices can be decomposed from the class. Frequent sequences are generated by joining the ID-lists of all pairs of atoms and checking the support of the resulting ID-list against minimum support. The ID-lists of the frequent sequences are kept as the set of atoms for the next enumeration. The process is repeated until all of the frequent sequences have been enumerated.

The cardinality of a sequence by joining the ID-lists of two sequences is called ”temporal ID-list join.” There are three possible types of join. The first one is joining the event atom with another event atom, like joining PB with PA and the possible outcome is PAB. The second one is joining the event atom with a sequence atom like joining PB with P→A and the possible outcome is PB→A. The third one is joining a sequence atom with another sequence atom, like joining P→A with P→B and possible outcomes are (P→A→B), (P→B→A) and (P→AB).

The new ID-list is created by checking the (sid,eid) pairs of both input sequences. To create an ID-list of the sequence (P→AB) the equality of (sid,eid) pairs of sequence (P→A) and (P→B) should be checked. The ID-list of the sequence (P→A→B) is computed by checking all (sid,eid) pairs (s_1,e_1) from sequence (P→A) and (s_2,e_2) from sequence (P→B) where s_1 = s_2 and $e_1 < e_2$, which means the item B follows the item A and pair (s_1, e_2) is added to the ID-list of the sequence (P→A→B).

Since the ID-list join is done by enumerating sequences in the class at each level, only the ID-list of last generated frequent sequences should be kept in the memory.

In [7] the performance of SPADE is compared with the performance of GSP [3]. The same synthetic datasets are used as those in GSP [3]. SPADE is twice as fast as GSP at lower values

Table 2.3: ID-list of Sequence P→A

Sequence	SID	EID
P→A	1	20
	1	30
	1	40
	4	60
	7	40
	8	10
	8	30
	8	50
	8	80
	13	50
	13	70
	15	60
	17	20
	20	10
P→B	1	70
	1	80
	3	10
	5	70
	8	30
	8	40
	8	50
	8	80
	11	30
	13	10
	16	80
	20	20

of support threshold, and the gap increases while decreasing the support threshold value.

2.2.1.3 Comparison of SPADE and GSP

GSP gives a general framework to find generalized association rules when the items have a taxonomy. With certain optimization techniques, the performance of the algorithms increased. SPADE uses a vertical layout dataset to calculate the support count, and uses an operation called temporal ID-list join. SPADE outperforms GSP since in SPADE only 3 database scans is performed, and an efficient method of calculating the support count of the sequence is used.

Table 2.4: Result of Temporal Id Join [7]

P→A→B

SID	EID
1	70
1	80
8	30
8	40
8	50
8	80

P→B→A

8	50
8	80
13	50
13	70

P→AB

8	30
8	50
8	80

The main advantage of GSP over SPADE is finding generalized association rules since SPADE does not take into account the presence of taxonomy.

2.3 Web usage mining on the Semantic Web

The number of studies that combine web usage mining and Semantic Web approaches is quite low. Most of these emphasize an automatic extraction of site ontology from Web content with minimal manual intervention.

One of the first studies of Web usage mining on the Semantic Web is [4]. This paper is composed of two parts. The first part is on extracting semantics from a Web page. The second is on the improvement of web usage mining using the semantics of a Web page.

In the first part, the ontology learning process is divided into two steps. In the first step, conceptual relations are discovered and relations between ontology concepts are generated.

Concept hierarchy is established using a knowledge acquisition method such as ONTEX (Ontology Exploration [30]). ONTEX relies on the knowledge acquisition technique of attribute exploration to extract all the concepts and their relevant combinations. ONTEX takes a set of concepts as its inputs and generates a hierarchy from them. This concept hierarchy together with the set of Web pages, is then passed to the second step, which is ontology learning. In this phase, the ontology is filled in and instances are extracted from Web pages.

In [4] it is suggested that the Web site log files should register the user behavior in terms of an ontology concept and ontology terms, and if the ontology is not presented, it can be extracted by the ontology learning explained above. A system for creating such semantic log files from a knowledge portal [11] has been described in [12]. It is suggested that this log then be mined by clustering or association rule mining algorithms.

Another related work is described in [5]. It is possible to examine this paper in two parts. The first part is about how the Semantic Web can improve Web usage mining, and the second part is about how usage mining can help to build up the Semantic Web.

In the first part, it is assumed that the web site logs contain every Web request by a session on one side and the ontology and knowledge base of the Web site on the other side. In the preprocessing step, each Web request is mapped to concepts at a chosen level of abstraction. The level of abstraction can be found dynamically by using an algorithm [3, 13], or it can be done manually.

After preprocessing, a sequence miner [15, 16, 17] or classfication data mining algorithm [18] [19] can be used to discover the sequential patterns in the transformed data. By sequential mining, common usage paths are extracted from Web logs.

In the second part of the paper, it is mentioned that, Web usage mining can be utilized to learn ontological structures and to classify instances. The navigational history of a session and manuel inputs of the visitor (i.e. search queries) can help to generate the conceptual structure. For example, user navigation after a search query can infer relatedness of the concept of the navigated page to the search query. This idea is used to extract the concept from the pages [20, 21, 22]. Furthermore, other techniques use a combination of concept extraction techniques (like keyword analysis) in order to generate recommendations [23].

In [6], a general framework has been presented for fully integrating domain ontologies with Web usage mining and personalization. In [6], the personalization process consists of three phases: data preparation and transformation, pattern discovery by data mining and recommendation. These phases are shown in figure 2.1. Among these phases, only recommendation is performed in real-time, the other phases are performed offline.

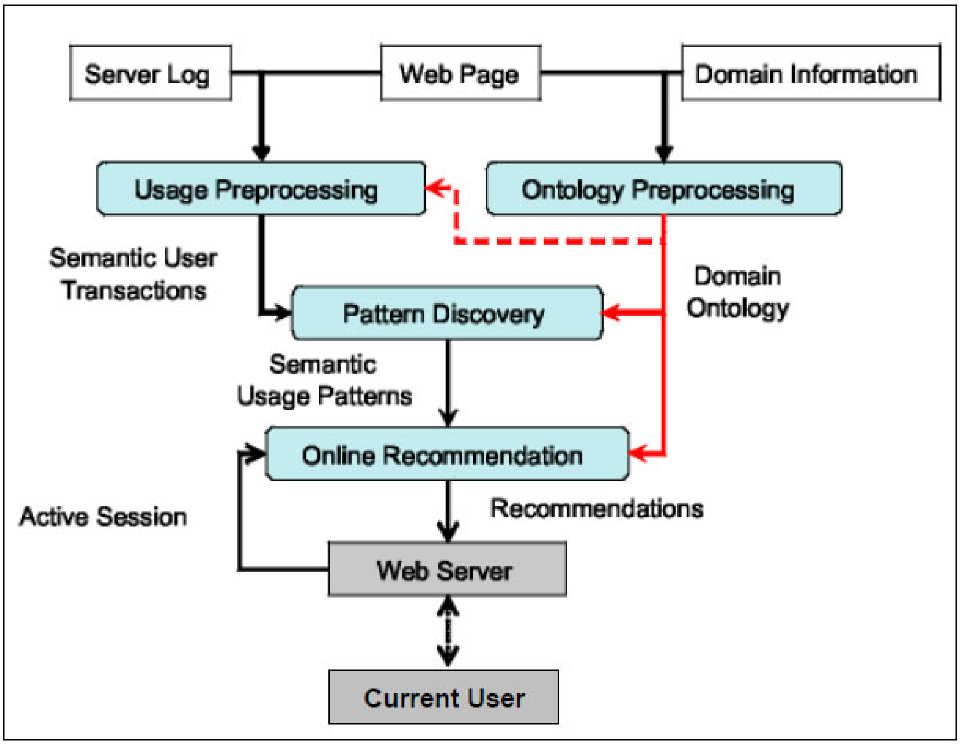

Figure 2.1: General Framework of Personalization. Depicted from [6]

The goal of the preprocessing phase is to transform users' navigational transactions into "semantic transactions" by mapping the accessed pages and resource to concepts and objects of the site ontology. The preprocessing is done by mapping usage-level instances to domain-level instances. To be more specific, instead of defining the user's navigational path as a set of Web pages and URLs, the transaction is represented by using instances from the knowledge base. Transforming to the concepts at different levels of abstraction provides more flexibility and lower computation cost in both the pattern mining phase and the recommendation phase.

In the pattern discovery phase, the aggregate representation of the patterns (users' navigational path) is calculated. The author uses clustering algorithms to find groups of sessions that are considered similar due to their common navigational patterns. The centroid of the transaction cluster acts as a representative of all of the transactions in that cluster. Since a concept in ontology can have many properties, the centroid calculation is performed independently for each attribute of the class. Since each attribute has a different data type and domain, a combination function must be defined for each attribute of the ontological concepts. A combination function is used for the calculation of "mean" and "average" functions when computing the centroid vector of the attributes.

In the recommendation phase, the recommender matches the current user navigation history against the discovered domain-level aggregate profiles. The usage profiles with matching scores greater than a certain pre-specified threshold are considered to represent the user's potential interests. The recommendation engine then recommends Web pages to the user according to the user's potential interest.

2.4 Recommendation Systems

Recommender systems [31] have become an important study area recently. The increasing amount of web content on web sites makes the recommender system an essential part of web sites. Recommender systems try to direct users to where they would like to go without getting the user lost in the huge amounts of information on the web site. The recommending of books, CDs and other products at Amazon.com [32] is an example of such a system.

The recommender's job is to produce a *rating* value between the user and the items that the user has not seen before. A better rating estimation leads to more successful recommendations and conversely, an incorrect rating estimation tends to produce irrelevant recommendations. The rating estimation is based on the underlying information, namely items previously seen by the user, the rating of items visited by the user, the collaborative rating of the items by all users, personal information of the user and so on.

Once the unknown ratings are estimated between the user and all items, actual recommendations of an item are made by selecting the item with the highest rating estimation, or alternatively N best items.

Many different methods from machine learning or data mining can be used to estimate the rating and, in fact, the recommender systems are generally named according to their approach to rating estimation. Recommender systems are classified into the following broad categories:

- Content-based recommendations: Similar items to the ones the user preferred in the past are generated as a recommendation.
- Collaborative recommendations: Items preferred by the people who have the similar taste to the user are generated as a recommendation.
- Hybrid approaches: The above recommedation methods are combined in this approach.

2.4.1 Content-Based Methods

In the content-based recommendation method, the rating of an item s for user c is estimated based on the user rating compared to the other items that are similar to s. In a movie recommendation, a user is recommended a movie according to the movies that the s/he rated highly in the past (e.g. the actor, director, genres). The movie that has the highest similarity to the preferred movies of the user in the past would be recommended.

There are several methods of finding the similarity of an item to a set of items. In particular, keyword analyzing techniques are applied to find recommendations. More formally, the algorithms analyze the content of the previously preferred and rated items for each word in the item and measure the closeness to the items that are not seen by the user. One of the best known is *term frequency / inverse document frequency* measure [33]

Content based recommendation assumes that there are some features of items in text format, like content or subject, or the features in text format are assigned manually to the item. This assumption limits the recommendation since some kinds of item such as multimedia data or graphical images, do not have text content. Another problem of content-based recommendation is *overspecialization*. That means, only items similar to those rated before by the user are recommended to the user, which limits the variety of items in the recommendation. For example, a user who has seen Al Pacino movies is generally recommended other movies with Al Pacino; the user never receives recommendation about other movies. In addition, *new users* would get irrelevant recommendations or would not get any recommendation at all since content-based recommenders need a substantial number of items that have been chosen or rated before by the user to understand the taste and preference of the user.

2.4.2 Collaborative Methods

Collaborative methods rely on the total preference and rating of *all users* instead of a single user's preference and rating. An example of collaborative recommender systems is the book and CD recommendation system at amazon.com [32]. Collaborative algorithms can be grouped into two classes: memory-based and model-based.

Memory-based algorithms make rating estimations based on all items rated by the user. The rating of an item for a user is calculated as the aggregate rating of all users who have similar taste and preference to the user, and who have also rated the item in the past. The most highly rated items are recommended to the user. Several aggregation methods are used to calculate the aggregated rating value; aggregation methods can be simple, like taking the average value, or they may be quite complex, including fuzzy and heuristic algorithms. The most common aggregation approach is weighted sum. In weighted sum, the ratings of the most similar users, contribute most to the aggregate sum and, on the other hand, the ratings of the users who do not have many common preferences with the user, contribute little to the aggregate sum. This approach brings another problem: that is the finding the similarity value between two users. The most common approach to finding similarity between two users is based on the ratings of items that both users have rated. In contrast to memory-based algorithms, the model-based algorithms are based on generating a model by using the previously rated items to generate the rating of the items in an automated fashion. Statistical and machine learning techniques are used to make the model.

Collaborative recommendation relies on the ratings of similar users. Therefore the similar users have to be found in order to generate recommendation. This task can be problematic for new users since they do not produce sufficient ratings to enable the finding of similar users. New items are not easily recommended to the users in collaborative recommendation since until the new item is rated by a substantial number of users, the new item will not be recommended to anyone. Another limitation of collaborative recommendation is the number of rated items. Generally, the number of rated items is relatively low when compared to the number of items to be rated. Therefore an effective recommendation system is needed, one which gives accurate ratings based on little underlying data.

2.4.3 Hybrid Methods

The hybrid approach combines the content-based and collaborative algorithms to make recommendations. Different methods are used to combine content-based and collaborative algorithm. These are:

1. Separately implementing content-based and collaborative algorithms and combining the prediction result.
2. Adding some content-based characteristics to the collaborative algorithm.
3. Adding some collaborative algorithm characteristics to the content-based algorithm.
4. Incorporating both algorithms and building a general framework.

All of these approaches are used to make hybrid recommendations.

CHAPTER 3

SYSTEM DESIGN AND IMPLEMENTATION

3.1 Overview

Web usage mining finds the common navigational patterns in terms of individual web page addresses and does not reveal the semantic meaning of the common navigational pattern. It is important to find the semantic meaning of these navigations since people generally navigate around the web site for the purpose of looking up a feature of an item, or just consider several features while navigating among various items. For example, on a movie site, some visitors navigate among the web pages only considering the casting of the movie for the purpose of watching a favorite star. Some people navigate only considering the target gender of the movie.

Common navigational patterns in terms of web page addresses only give a sequence of pages that shows which page is more visited than others, and does not give the precious semantic information about the purpose of frequent navigation paths. For example, a frequent navigation path in terms of web page address reveals that, people who visit the movie ”GodFather”, then visit the movie ”Taxi Driver”; also people who visit ”Scarface” also visits ”Goodfellas”. In fact, people who visit movies starring ”Al Pacino” , then visit movies starring ”Robert De Niro” and this information cannot be revealed by classical web usage mining.

In this thesis, by including semantic information in web usage mining, semantic frequent navigations will be extracted. This task will be done by using the ontology of the web sites. Domain ontology contains many classes and these classes may contain many properties (i.e.

attributes or features; for example movie class has casting, director, and target gender features). For each property of the classes, the common frequent navigation patterns will be extracted. The results will be used to make recommendations to the users.

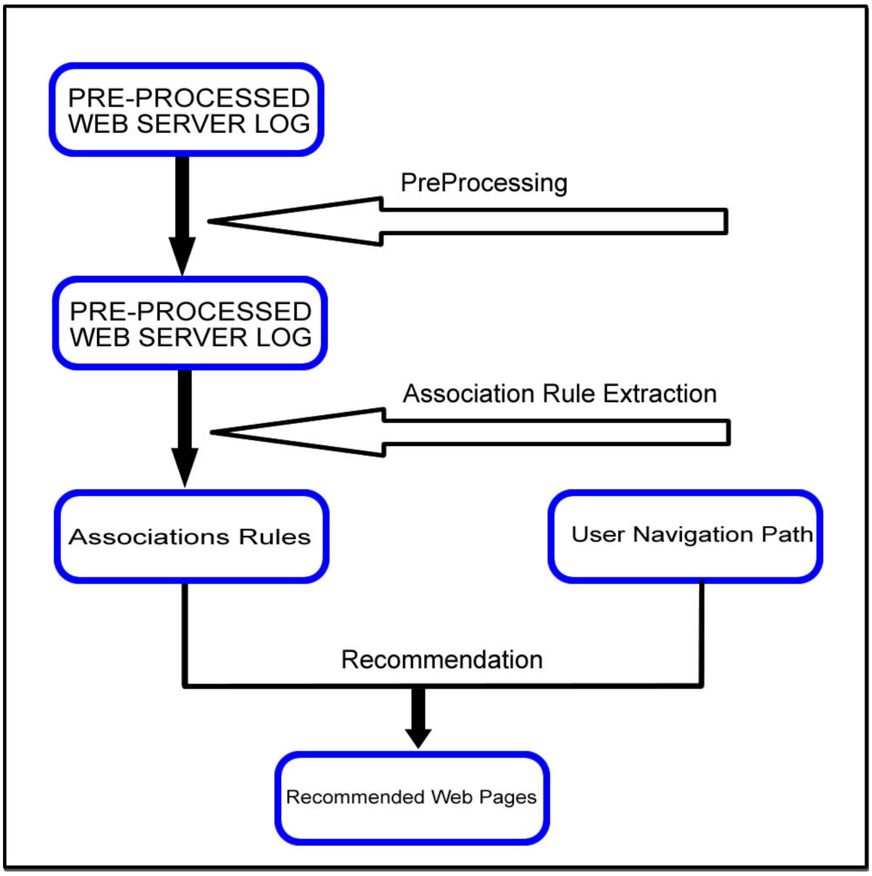

Figure 3.1: General Framework of Overall System Design

As shown in Figure 3.1, the proposed system consists of 3 phases. The first phase is the preprocessing step. Generally, association rule mining algorithms assume that the dataset is pre-processed before pattern generation. However, in the real world, the dataset comes in raw form, requiring pre-processing. The Web server log files contain noisy and irrelevant visit logs. Therefore, a preprocessing step should be designed before starting the finding of association rules. In this thesis, a new pre-processing step is designed from scratch, which is quite an interesting topic for further research.

In the preprocessing step the unnecessary entries from web server logs are pruned and the navigation histories of each visitor of the web site are extracted. The navigation history is constructed as a series of ontology individuals instead of a series of web page addresses.

The next phase is the association rule extraction step. There are several algorithms with which to generate association rules. Two of the most popular algorithms are SPADE and GSP, which were described in the previous chapter. The main difference between these two algorithms is that GSP generates generalized association rules, i.e. it allows for the presence of the taxonomy. However, SPADE does not. Taxonomy can be encountered in many ontology definitions, since items of some classes would have a parent-child relation (i.e. is-a relation) among them; therefore, an algorithm that has the ability to find generalized association rules is a necessity for the proposed system. On the other hand, GSP is slower than SPADE.

In the design phase of the system, the use of the GSP algorithm was planned, if the individuals of the ontology class do have a taxonomy, since it allows the establishment of generalized rules; on the other hand, if there is no taxonomy, the SPADE algorithm would be used due to its efficiency. However, during tests on large datasets, it was revealed that GSP was extremely slow in comparison to SPADE. Therefore the system was re-designed, and SPADE extended with the ability to generate generalized rules under a given taxonomy. In Appendix B, there is a detailed time comparison of both algorithms.

In particular, the vertical dataset construction step of the SPADE algorithm was changed to allow the production of generalized association rules. In addition, the sequence - subsequence relation test mechanism of SPADE was changed to take taxonomy into account.

Moreover, in the evaluation it is revealed that there are some uninteresting frequent sequences which are not mentioned in the original paper. Uninteresting frequent sequences donot make any contribution to the generation of association rules, and removing uninteresting frequent sequences does not alter the quantity and quality of the association rules. Therefore, the pruning of uninteresting frequent sequences was added to the SPADE algorithm. Hence, the association rule generation becomes more efficient since many frequent sequences are eliminated before starting the creation of association rules.

The third and last phase is the recommendation phase, in which the association rules are joined with the user's online navigation path, and new pages are recommended to the user. Briefly, in recommendation, several most recently navigated items are taken as a search pattern. All the association rules are scanned, and according to the rules matched with the search patterns, several web pages are recommended to the user.

The success of the system is measured by evaluation of the recommendations. Evaluation is made by performing 10-fold cross-validations on 3 datasets. The precision and coverage values of the recommendations are measured for each session in the evaluation dataset. Moreover, two new measurements that are called precision-with-threshold and coverage-with-threshold are introduced. By changing certain parameters and keeping the other parameters constant, many tests are performed on datasets, especially on the SECKIN web site.

In experimental work, two real-world Web server logs are used. The first one is the visit log of the book store Web site http://www.seckin.com.tr (SECKIN). The other one is the Web log of the Computer Engineering department (CENG) of the Middle East Technical University.

The number of rule selection criteria is added to the recommendation to evaluate time and recommendation accuracy relationship in terms of the number of rules. The rule number selection defines the number of association rules to be used in the recommendation phase. A lower number of association rules leads to a lower number of, but faster recommendations. On the other hand, high numbers of association rules lead to slower algorithms, but produces more recommendations.

In order to measure the correctness of recommendations, an evaluation methodology is used. The dataset is divided into training and evaluation/test parts. Association rules are extracted from the training dataset and the accuracy of the recommendation is measured on the evaluation dataset. In many recommendation papers, precision and coverage values are calculated to evaluate recommendation accuracy. In this thesis, two new evaluation measurements are added called precision-with-threshold and coverage-with-threshold. These values help to find how the precision and coverage are distributed, and they measure the success of the precision and coverage values with respect to a given threshold value (i.e. acceptance level).

The following section describes details of the system design and implementation. It concerns the design of the system, including how the project is designed to carry out Web usage mining with semantic information. In the implementation section, technology, the design of the classes, and a snapshot of the implemented system is given.

3.2 System Design

3.2.1 Preprocessing

Before the search for association rules commences, the Web server log files are processed to be read for the SPADE algorithm as shown in the Figure 3.2. In this step, Web server log files are pruned, transactions are extracted, and ontology class individuals are mapped to the Web page addresses.

This step is the most time-consuming since the volume of the Web server logs is generally huge. It can take hours or days, depending on the Web server log file size. Therefore, it is wise to store the pruned result in a permanent storage area like a database to avoid having repeating the process.

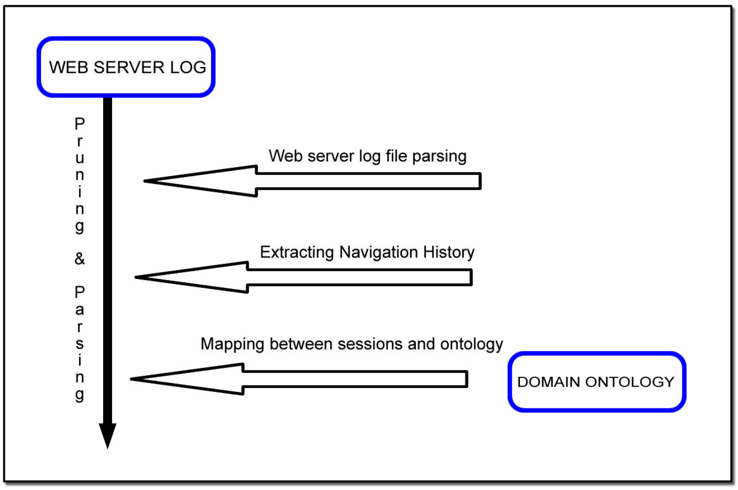

Figure 3.2: General Framework of Pruning and Parsing

3.2.1.1 Parsing Web Server Logs

The Web server registers all the requests made to the server in a log file. Browsers make requests to get a file from the Web server to display the content in the browser. Any type of file can be requested from the Web server, as long as it is allowed to serve such a file. The requested object may be a static Web page, a dynamically generated Web page, a picture or a cascading style sheet file.

Table 3.1: Generally Requested Files From Web Servers

File Definition	Extensions
Static Web Page	.html .htm
Dynamic Web Page	.asp .aspx .php .jps
Web Servise Files	.asmx .svf
Picture and Figures	.gif .jpg .jpeg .png
Multimedia	.wma .flv .swf .mp3
Cascading Style Sheet	.css
Script File	.js
Metadata File	.owl .rdf .xml
Document	.pdf .doc
Archive	.zip .rar

Some of the file types a Web server may serve are listed in table 3.1. The list is not complete: there are many more file types that can be requested from Web servers. Web servers keep the request time, requested Web addresses, request type (like GET, POST, HEAD), http version, user agent information, client IP address, and response status (e.g. 200 is response successfully delivered, 404 requested file not found, 500 internal server error) and referrer address. In addition, some Web servers can gather additional information of regarding the request, such as the port number, cookie information, and so on.

Each request is written as a new line to the log file. Hence, the number of lines in the log file is equal to the number of requests to the Web server. The size of the log file can dramatically increase when the number of requests increases. For example, if a Web site gets 250,000 requests per day, its daily log size will be around 100Mb.

Table 3.2: Sample Web Server Log File

Time	Method	Address	Client IP	Version	User Agent
04:56:25	GET	/urun.aspx?productID=26	74.6.27.34	HTTP/1.0	Mozilla/5.0+...
04:56:54	GET	/urun.aspx?productID=28	74.6.27.34	HTTP/1.0	Mozilla/5.0+...
04:58:12	GET	/urun.aspx?productID=29	74.6.27.34	HTTP/1.0	Mozilla/5.0+...

Table 3.2 shows sampled visit log of date 2008-01-20 from the Web server log file used in the thesis.

The first phase of the log file parsing is the pruning of the non-responded Web requests. Non-responded requests can be detected from the status field of the request log entry. The status of the successfully responded Web requests is 200. If the request has some other status, like 404 or 500, that means the requests were not responded to properly. On average, 5% of the Web requests are non-responded. In case of a bug in the web application, or when the server is busy, the percentage of non-responded requests increases dramatically.

The second step of log file parsing is the elimination of the requests made by certain software agents which sometimes automatically request Web content from a Web site. The most common software agents are Web crawlers, which are also called Web spiders or Web robots. Web crawlers are mainly employed by search engines and used to get all visitable pages from the World Wide Web to index for search requests. To keep up-to-date data, Web crawlers may crawl a Web site many times a day, according to the change rate of the Web site. Therefore, Web crawlers make a large portion of the visit logs. It has been calculated from the experimental datasets that 30% of all Web requests are made by Web crawlers.

Table 3.3: Web Crawler User Agents

Agent	Search Engine
Mozilla/5.0+(Googlebot/2.1;++http://www.google.com/bot.html)	Google
Mozilla/5.0+(Yahoo!+Slurp;+http://help.yahoo.com...)	Yahoo
msnbot/1.0+(+http://search.msn.com/msnbot.htm)	MSN Search
Mozilla/5.0+...+Gecko/20071127+Firefox/2.0.0.11	Firefox 2.0 Web Browser
Mozilla/4.0+(+MSIE+7.0;+Windows+NT+5.1;)	I.E. 7.0 Web Browser
Mozilla/4.0+(+MSIE+6.0;+Windows+NT+5.1;)	I.E. 6.0 Web Browser
Mozilla/5.0+(Macintosh;+U;+Intel+Mac+OS+X;+en)+Safari/419.3	Safari Web Browser
Opera/9.24+(Windows+NT+5.1;+U;+tr)	Opera Web Browser

Since Web crawlers do not portray the common interests of the visitors, requests from Web crawlers are excluded. When a software agent makes a request to the Web server, it gives information about the software agent in the request body. This definition is kept in the user agent field in the log files. Table 3.3 shows some of the Web crawlers' user agent's definitions.

The third step when pruning log files is the removal of irrelevant requests from the log file. Non-Web page requests, such as an image request or style sheet request are not taken into account since these files are auxiliary files for displaying Web site to the user. Within the scope of this thesis not all the pages of the overall Web site are included. In the final set, association rule mining runs on a subset of the Web pages. For example, only the catalog pages and the product pages are included. The subset of the Web pages to be taken into consideration is decided according to the target of the mining. For example, if the relationships between books are to be extracted, then only the book Web pages are included. In addition to this, pages specific to some actions, such as payment or contact, are eliminated as well.

Extraction is done by reading the Web server log line by line. Each line is parsed into the fields, e.g. client IP, request address etc. The order of the fields is consistent on a given Web server but it can change from server to server. The order of the fields is mostly written in the documentation of the Web server and some of the Web servers allow changes in this order.

After fields are extracted, pruning is done according to the value of the fields. Status is used to prune non-responded requests, address fields are used to prune with respect to the Web page address, and user agent fields are used to prune Web crawlers. Mostly, large volumes of the Web log are discarded, for example, 88% of the web requests are eliminated in the pre-processing phase of the SECKIN Web site.

3.2.1.2 Extracting Navigation History

The next step after pruning is the extraction the navigation history of each session from the log file. The navigation history is the set of Web objects requested by the user in his active session time. A session is established when the end-user makes the first request to the Web server, and the session is torn down after a period of idle time from the end-user. The allowed idle time is called the session timeout. Most Web servers have 20 minutes of session timeout. If the same end-user makes another request after the session is purged, a new session is created for him. If an end-user makes a request within his session-time, the idle-time timer resets, and his session continues.

Hypertext Transfer Protocol (HTTP) is a stateless protocol so the server does not know about the browser's past navigation history, or anything done in the past by the user. Therefore, an original http protocol cannot support this type of session. To support such a session, some session management techniques are used by Web developers.

Generally, the Web servers store the session information in the memory of the Web server, and this information is accessible by an identifier called Session-ID. Session-ID is generated at the first request from the end user. It is stored in the end user and at each request, this Session-ID is sent to the Web server using various kinds of information exchange techniques, such as browser cookies, query strings, or a hidden field in the Web page. Most Web servers do not guarantee global uniqueness of Session-IDs; however, they guarantee the uniqueness of Session-IDs among active sessions.

To extract the true session-visit history, the Session-ID of each request should be known. However, the Web servers do not keep any information related to the Session-ID in the Web site log. But regarding session management techniques, some clues can be found in the Web server log file about the Session-ID of the request.

Some Web servers keep logs of cookie information. If the Session-ID is kept in the cookies then the session information can be found in the Web server log. However, the Session-ID information is only read from the cookies of the second request, since in the cookies of the first request there is no Session-ID value, after the first request, a new Session-ID is generated and written into the cookies. Hence, the first request of the session should be extracted from the Web server log and is added as the first visit of the navigation history for the session. It is done by finding the last request prior to the first request of the session with the Session-ID cookies value by matching the request's user agent field and client IP addresses.

Table 3.4: Some of the requests cookie information

Cookie Information
n/a;+ASP.NET$_{S}essionId = rgbkun55by551o45ras542r1$;
n/a;+ASP.NET$_{S}essionId = nmkotiegy2zp13by11nfprfa$;

Some Web developers use query strings as the information exchange mechanism for Session-ID, especially when end-user browsers do not allow cookies. Since the requested address is found in the Web server log, the Session-ID information can be extracted from the requested URL.

Some Web developers use hidden fields on the Web page as the Session-ID exchange mechanism. Since the response content is not kept in a log file, there is no way to find the Session-ID information from the server log if such a session management technique is used. In addition, some Web developers do not use Session-IDs since they do not need any information about the end-user. In this case there is no way to find session information from servers.

If there is no information about the Session-ID in the Web server log files, then the session information is extracted programmatically by using the fields in the Web server log related to the client. The client information that is useful in extracting sessions are time, user agent and IP address.

The unique Session-ID is given to the first request of the Web server. The end of the session is the time of the request. Among the following requests, if there is one whose user agent and IP address is the same as the request under consideration, and whose request time is not greater then the end of the session, plus the session time-out period, then that request is added to the session, and the end of the session will be the time of the last added request. If the following request does not belong to the session, then a new Session-ID is generated and the same task is repeated for this request. This process is repeated until Session-IDs of the all requests are defined. This process is expressed in Algorithm 3.2.1.

Algorithm 3.2.1: ExtractSession($\mathcal{L}$, *session_timeout*)

---

comment: Extract sessions from a given Web server log file.

$T = \emptyset$;

$Session = \emptyset$;

for all log entry $\beta \in \mathcal{L}$

do for each $T_i \in T$

do
$$\begin{cases} \textbf{if } T_i.ip = \beta.\text{ip and } T_i.user\_agent = \beta.\text{user\_agent and } T_i.lasttime \leq \beta.\text{time + session\_timeout} \\ \quad \textbf{then } \text{Session } = T_i; \\ \quad \textbf{else} \\ \quad \textbf{then } \begin{cases} \text{Session = new Session;} \\ \text{T = T} \cup \text{Session;} \end{cases} \\ \text{Session = Session} \cup \beta; \\ \text{Session.lasttime} = \beta.\text{time;} \end{cases}$$

There is a drawback to this algorithm. There can be many sessions that are using the same browser from the same IP address, and these sessions are calculated as a single session by the session parser algorithm. Generally, companies or big institutions use a single static IP address when accessing the Internet. Therefore, all the users in the company or the institution that use the single static IP to access Internet will be displayed as a single session. Although browser distinction is used, the number of commonly used browsers is very small, so the browser distinction is not very helpful in separating sessions.

The SECKIN Web site log file contains the Session-ID information in the cookies field of the request. Therefore, sessions are extracted from Web site log file by parsing the cookie field. The log file of the CENG Web site does not contain any Session-ID information; hence, the sessions are extracted from the log by using IP addresses, user agent information and the time of the request.

Since finding the session navigation history is a time-consuming job, the result is stored in the database.

3.2.1.3 Mapping between sessions and ontology

The next step of pre-processing is to map between ontology individuals and the requested Web address in the Web server log. The Web server does not register semantic information about the request in the log file, only the address of the request. Therefore, before starting the frequent sequence finding, mapping between ontology and the Web site address should be carried out.

To include semantic information on a Web page, there should be an ontology which defines the classes of the domain space and their properties showing the relationships among them. Semantic information Web ontologies are written by a knowledge representation language. Web Ontology Language (OWL) and Resources Definition Language (RDF) are endorsed by the World Wide Web Consortium. OWL has more expressive power over RDF. Hence, OWL is generally used as the ontology language for Semantic Web.

The two real Web sites that are used in this thesis did not contain semantic information since their ontologies were not defined. Ontologies have been defined for both of them. The OWL definition of these Web sites can be found in Appendix C. There are five classes in the SECKIN Web server logs which are Book, Author, Category, Vendor and PublishYear. There are three classes of the CENG Web site ontology, which are Author, Group, and Thread. Each class has many individuals, e.g. computer, and education are the individuals of class Category. These individuals are kept in a separate ontology instance file. These ontology files can also be found in Appendix C.

A sample request address for the SECKIN Web site log is shown in Table 3.5. The address and

Table 3.5: Sample Request Address

Cookie Information
/urun.aspx?productID=10653
/urun.aspx?productID=10706
/urun.aspx?productID=10763
/urun.aspx?productID=10766
/urun.aspx?productID=10782
/urun.aspx?productID=10801
/urun.aspx?productID=10808
/search.aspx?key=analiz&page=6
/search.aspx?key=kpss+a+grubu
/search.aspx?key=kpss+genel+yetenek
/search.aspx?key=kpss+meslek
/reyon.aspx?catid=210000
/reyon.aspx?catid=210000&page=1
/reyon.aspx?catid=210000&page=18

ontology individuals mapping is stored in an ontology file called the mapping file. In the mapping file the ontology instances are mapped to the Web address. There can be more than one corresponding individual for a Web address. For example /urun.aspx?productID=10763 is a Web page about books, and it contains one individual from the classes Book and PublishYear, but it may contain one or more individuals from the class Author . /reyon.aspx?catid=210000 is a Web page of a book catalog; therefore, it contains around 10 individuals of the class Book, PublishYear, and 10 or more individuals of the class Author.

The concept ontology files, instance ontology files, and mapping ontology files are created by Protégé[38]. Protégé is developed by Stanford University in collaboration with the University of Manchester, and it is a free and open source ontology editor system.

Table 3.6: Navigation paths in terms of the individuals of PublishYear class

Session-ID	PublishYear Individuals
923	1996 1997 2001 → 1996 → 1996 1997 2002 2003
624744	2008 → 2007 → 2008 → 2000,2001,2003,2004

For each class of the ontology, all sessions and their visited pages are extracted. For each visited address, its corresponding instances are extracted from mapping ontology files. For each concept, a different session-visited instance transaction database is constructed. For example, Table 3.6 shows sessions in terms of the individuals of the PublishYear class.

The ontologies of both sites we have used are relatively simple domain ontologies. There is no sub-class relation between classes, and no complex properties of the classes are defined. However, there is a taxonomy relation among the individuals of the Category class in the SECKIN Web site ontology. In such straight-forward ontology definition, another option can be used to map ontology individuals to the Web site address, which is to read individual values directly from the database.

Generally, dynamic Web sites have a relational database to store their data. The ontologies classes of such Web sites are generally synchronized with the design of the database and ontology individuals are synchronized with the data in the database. Most of these Web sites use an automatic program to extract ontology from the database from time to time. In such circumstances, instead of reading the ontology classes and individuals from the ontology files, the classes and individuals can be directly extracted from the database. It is obvious that when data is read from the database, many capabilities of ontology cannot be used. In particular, neither reasoning nor inference can be made on such a simple ontology.

It is beneficial to store the navigation paths in terms of individuals in a permanent storage area since, while calculating frequent items, this dataset is needed many times, and the construction of the navigation paths in terms of ontology individuals is a highly time-consuming task.

3.2.2 Frequent Navigation Pattern Generation

After preprocessing, the next step is the extraction of frequent navigation patterns. In this step, frequent sequences are extracted from the transactions. SPADE is used as the association rule mining algorithm. To make it possible to extract generalized association rules from SPADE, some extensions are made to it.

As the first step to finding association rules, vertical datasets are constructed from the preprocessed data. After that, as described in the SPADE algorithm, frequent 1-sequences, 2-

sequences, and 3-sequences and longer are found.

SPADE is extended and some optimizations applied to make the algorithm more efficient. The most important extension is the addition of the capability to generate generalized association rules. The method that finds whether a sequence is a sub-sequence of another sequence and the vertical dataset construction phase of SPADE are changed.

As an optimization, a new step called *pruning uninteresting frequent sequences* is added to the SPADE which prunes the uninteresting frequent sequences and hence makes the subsequent steps more efficient. Also, a new method for finding frequent 2-sequences is included, which makes the process faster if the number of frequent 1-sequences is low.

3.2.2.1 Construction Vertical Dataset

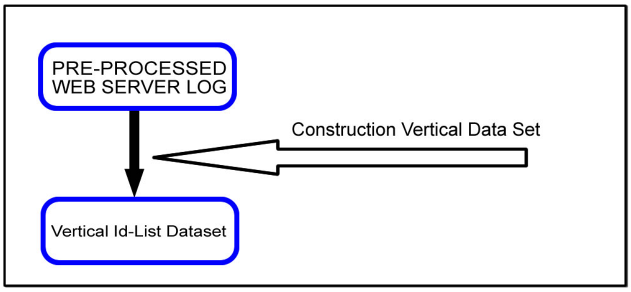

Figure 3.3: General Framework of Vertical Dataset Construction

As the first step of vertical dataset construction, all distinct ontology individuals are extracted from the ontology. These distinct items are stored in a list data structure in the memory. The general framework of vertical dataset construction is shown in Figure 3.3.

In the second step, all the pairs (Session-ID, Event-ID) for each item are extracted and written into a data structure that stores the vertical dataset. The database table that is used to store the result of the preprocessing step is used to read (Session-ID, Event-ID) pairs to construct a vertical dataset.

If the taxonomy exists, then a further step is performed, in which the taxonomy information is written into a tree data structure. This data structure keeps the *is-a* relations between class individuals. The taxonomy is read from the instance ontology file. Moreover, extension to the vertical dataset is performed, which is each (Session-ID, Event-ID) pair of an item is also added to the item's taxonomical parent. For example Computer Engineering is a sub-category of Engineering so that each (Session-ID, Event-ID) pair from Computer Engineering also belongs to the Engineering category.

In addition, the minimum support count is calculated at this step. This is done by multiplying the minimum support threshold percentage with the number of distinct sessions in the database. The number of distinct sessions is found by enumerating each (Session-ID, Event-ID) pair in the vertical dataset.

The following section describes the phases of the common navigational pattern finding, as shown in Figure 3.4. The next phase is finding the frequent sequences that consist of one item (1-sequences).

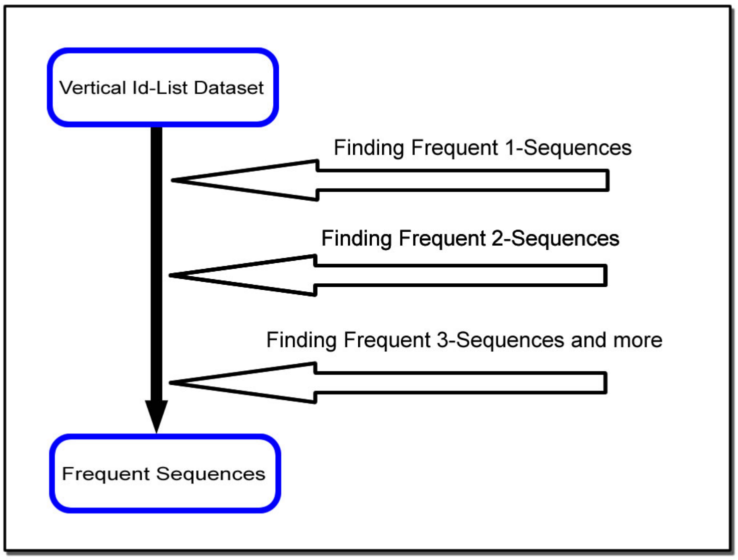

Figure 3.4: General Framework of Common Navigational Pattern Finding

3.2.2.2 Finding Frequent 1-Sequences

Finding 1-sequences is a relatively a straightforward step. For each item its ID-list (i.e. Session-ID, Event-ID) pairs are read from the vertical dataset and the number of distinct Session-ID is calculated and if this count is greater than the minimum support count then this item is added as frequent 1-sequence.

Algorithm 3.2.2: FIND-1-SEQUENCES($\mathcal{V}$, *minimum_support*)

---

comment: Find 1-sequence from given Vertical Dataset

$\mathcal{T} = \emptyset$;

for each item I $\in \mathcal{V}$

$$\textbf{do} \begin{cases} \sigma\ (\mathrm{I}) = \{ \text{ distint sid in } < \text{sid, eid} > \text{ pairs of ID-list(I) } \}; \\ \textbf{if } \sigma\ (\mathrm{I}) \geq \mathit{minimum\_support} \\ \quad \textbf{then } \mathcal{T} = \mathcal{T} \cup I; \end{cases}$$

3.2.2.3 Finding Frequent 2-Sequences

In [7], finding frequent 2-sequences is not explained in depth since [7] emphasizes its distinct approach to generating 3 and longer sequences. [7] does not restrict the developers to using any methods to find the frequent 2-sequences, but suggests two methods and leaves the choice to the developer. These are performing a preprocessing step to find the support count of all combinations of 2-sequences and converting the vertical dataset to a horizontal dataset on the fly, and finding the support count of all 2-sequences presented in the horizontal dataset.

In the proposed system, the method of converting the vertical dataset to a horizontal one is used for finding frequent 2-sequences. However, in some test case, it was revealed that the conversion and scanning of the horizontal dataset take too much time. It was further revealed that the number of frequent 1-sequences is small. Therefore, for such circumstances, a new method was implemented, which is the joining of frequent 1-sequences with themselves on the fly.

Converting to a horizontal dataset is a rather straightforward task. Each item from the vertical dataset is read with its ID-list, i.e. (Session-ID, Event-ID) pairs. After that, for each distinct Session-ID, a new list is created. If the list has already been created, then the item is appropriately placed on the list. The items in the list are ordered by event timestamps.

Algorithm 3.2.3: CONVERT-TO-HORIZONTAL($\mathcal{V}$)

---

comment: Conversion to horizontal dataset from a given vertical dataset

$\mathcal{T} = \emptyset$;

$\mathcal{H} = \emptyset$;

for each item I $\in \mathcal{V}$

do for each { distint sid in $< sid, eid > \in$ ID-list(I)

$$\textbf{do} \begin{cases} \textbf{if } \text{H contains } Session_{sid} \\ \quad \textbf{then } \begin{cases} Session_{sid}.put(I); \end{cases} \\ \quad \textbf{else } \begin{cases} H = H \cup Session_{sid}; \\ Session_{sid}.put(I); \end{cases} \end{cases}$$

After the construction of the horizontal dataset, each session is enumerated one by one, and each pair of items, without considering the gap between them, is added as candidate frequent 2-sequences along with its occurrence count as 1. If the pair is already in the list, then its occurrence count is incremented by 1. After all iterations are finished, all candidate frequent 2-sequences whose occurence count is greater than the minimum support count are added to the frequent sequences.

Algorithm 3.2.4: FIND-2-SEQUENCES($\mathcal{H}$, $minimum\_support$)

comment: Finding 2-sequences from horizontal dataset

$Occurence = \emptyset$;

$Candidates = \emptyset$;

for each Session S $\in \mathcal{H}$

do for each $< I_i, I_j > \in$ S

do if $< I_i, I_j > \in Candidates$

then $\left\{ Occurence[< I_i, I_j >] = Occurence[< I_i, I_j >] + 1; \right.$

else $\begin{cases} \text{Candidates} = \text{Candidates} \cup < I_i, I_j >; \\ Occurence[< I_i, I_j >] = 1; \end{cases}$

The other way to calculate frequent 2-sequences is by joining all the frequent 1-sequence with themselves and counting the cardinality of the result. For example, if A and B are frequent 1-sequences, then the sequence {A $\rightarrow$ B } , {B $\rightarrow$ A} and {AB} are the candidate frequent 2-sequences. The cardinality is calculated by the temporal join operation described in [7].

The first way is more complex, but is a more efficient algorithm for most scenarios. However, if the number of frequent 1-sequences is low, and the average number of items in the transactions is high, then the performance of the first way overly degrades, and the second way becomes more efficient.In this thesis, the first way is used while calculating the 2-sequences of the SECKIN Web server log, while the second way is used for the CENG Web server log.

3.2.2.4 Finding Frequent 3-Sequences and more

The next step is finding 3-sequences and over by using the frequent 1-sequences and 2-sequences. The efficiency of SPADE is revealed when computing frequent 3-sequences and over. For each item$_i$, an equivalence class is composed from the frequent sequences whose first item is equal to the item$_i$. When the class is first constructed, there is only a single 1-sequence, and one or more frequent 2-sequences. The atoms of an equivalence class are a set of frequent sequences in the class found in the last iteration. In the first iteration, the atoms are the frequent 1-sequences plus frequent 2-sequences. For each iteration, all the sequences in the atoms are joined by a temporal ID-list join operation. The temporal ID-list join operation produces one or more new candidate sequences along with their ID-list. If the support count of the candidate frequent sequence is greater than the minimum support count, then the candidate sequence with its ID-list information is added to the class as a new atom for the next iteration as well as to the list that keeps all the frequent sequences. The newly added frequent sequences to the equivalence class make up the atoms for the next iteration. The process repeats until all the atoms are joined and no new frequent sequences are found.

The joining of the atom sequences deserves more explanation. Before the joining of two atom sequences, the suitability of the join operation between two atoms is checked. To join two sequences at most, one item of the sequences should be different and this item should be the last item of either sequences; alternatively the sequences should be the same. The next step is finding whether the operation is an event join or a sequence join. According to the operation type, the join operation is made as described in [7].

While joining the atoms, breadth-first or depth-first search algorithms can be used. A depth-first search is more memory efficient than a breadth-first search, but is more complex. If there is a memory shortage, a depth-first search can be used. At first, breadh-first search was used in this thesis. However, after encountering a memory insufficiency, a depth-first search was used.

Algorithm 3.2.5: ENUMERATE-FREQUENT-SEQ($\mathcal{S}, min\_sup$)

comment: Frequent sequence generation from the equivalence class

for all atoms $A_i \in \mathcal{S}$

do $\begin{cases} T_i = \emptyset; \\ \textbf{for} \text{ all atoms } A_j \in \mathcal{S} \\ \quad \textbf{do} \begin{cases} \mathcal{L}(R) = \mathcal{L}(A_i) \cap \mathcal{L}(A_j) \\ \textbf{if } \sigma(R) \geq min\_sup \\ \quad \textbf{then } T_i = T_i \cup R; \mathcal{F} = \mathcal{F} \cup R; \end{cases} \\ \text{Enumerate-Frequent-Seq}(T_i, min\_sup) \end{cases}$

If there is enough memory to store all the frequent sequences and their ID-lists, then the operation can be done in the memory. However, there is a limited memory, and sometimes not all the ID-lists of the intermediate sequences fit in the memory. Under that circumstance, the equivalence class can be decomposed into smaller equivalence classes. For example, {A, AB, ABC , AC, ACD , ACE, AD} forms a 1-equivalence class. {A} is the root sequence of the class. If the class needs to be decomposed into smaller classes, then 2-sequences is going to be the root sequence of the decomposed classes. The new classes are {AB , ABC}, whose root is the sequence {AB},{AC , ACD , ACE}, whose root is the sequence AC and {AD}, whose root is the sequence AD. Therefore, through decomposing, the size of the class will never exceed the size of the memory.

3.2.2.5 Frequent Sequences Pruning

This phase is not mentioned in the original paper but added as an optimization to SPADE.

After generating frequent sequences, some of them can be uninteresting, namely those whose support can be known from other sequences.

Generally, a new frequent sequence is added to the list of frequent sequences for all combinations of items in an event. For example, from Table 3.7 sequences {A → ABCDE → AB } , {A

$\rightarrow$ ABCD $\rightarrow$ AB }, and {A $\rightarrow$ ABC $\rightarrow$ AB } are the same since all sequences have a sub-event of the event {ABCDE}, and they all have same support count. Therefore, all sequences except the biggest are pruned.

In the table, the sequence { A $\rightarrow$ ABCDEF $\rightarrow$ AB } is not pruned since its support count is different to that of the pruned sequences. Also, the sequence {A $\rightarrow$ ABF $\rightarrow$ AB} is not pruned since the event {ABF} is not a sub-event of the {ABCDE}.

Table 3.7: Sample Frequent Sequences

Sequence	Support Count	Pruned
A $\rightarrow$ ABC $\rightarrow$ AB	10	yes
A $\rightarrow$ ABCD $\rightarrow$ AB	10	yes
A $\rightarrow$ ABCDE $\rightarrow$ AB	10	no
A $\rightarrow$ ABF $\rightarrow$ AB	10	no
A $\rightarrow$ ABCDEF $\rightarrow$ AB	9	no

The pruning starts with sorting all the sequences by their item counts. For each sequence, if the support count and event count are equal to the support count and event count of the sequence next to it, and one of them contains the other, the smaller one is pruned from the frequent sequences. Pruning frequent sequences prevents the generation of too many rules and makes the evaluation more efficient.

Algorithm 3.2.6: PRUNE($\mathcal{S}$)

comment: Prunes uninteresting frequent sequences

$Sort(\mathcal{S})$;

for each $S_i \in \mathcal{S}$

do if ISUNINTERESTING(S_i, S_{i+1}

then $S = S - \{S_i\}$

procedure ISUNINTERESTING(S_i, S_j)

if $EventCount(S_i) = EventCount(S_j)$ and $\sigma(S_i) = \sigma(S_j)$ and $S_i \preceq S_j$

then return $(true)$

else return $(false)$

end procedure

3.2.2.6 Association Rule Extraction

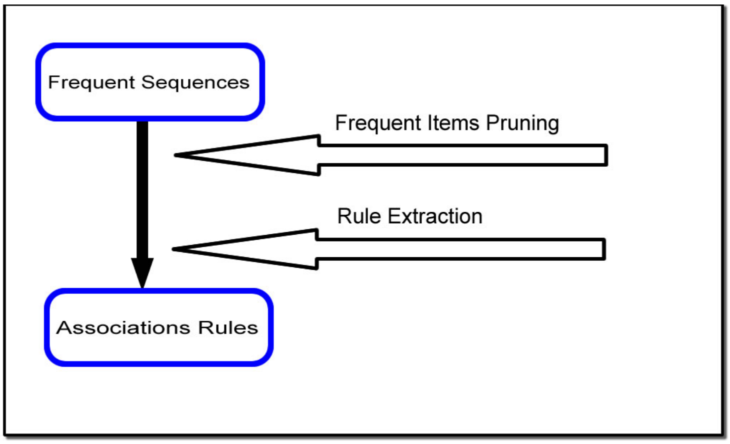

Figure 3.5: General Framework of Association Rule Extraction

The association rules are generated after all the frequent sequences are found and pruned as shown in Figure 3.5. The association rules are the main element of the recommendations. Therefore, it is necessary to generate the rules before making any recommendation.

Rules are found by comparing two frequent sequences. For example, { A → B } and {A → B → C } are frequent sequences generated before, and their support counts are 10 and 5 respectively. Sequence 2 starts with Sequence 1; therefore, a rule can be extracted by comparing these two sequences. The rule { A → B } → { C } is extracted from the sequences. The confidence of a rule shows the strength of the rule and is calculated by dividing the support of sequence 1 by that of sequence 2. Thus, the confidence of the rule is 0.5.

Algorithm 3.2.7: RULEGEN($\mathcal{F}$, min_conf)

---

comment: Generates rules from a given frequent sequences

for all frequent sequence $\beta \in \mathcal{F}$

do for all subsequence $\alpha \leq \beta$

do $\begin{cases} conf = fr(\beta)/fr(\alpha) \\ \textbf{if } conf \geq min\_conf \\ \quad \textbf{then } \text{output the rule } \alpha \Rightarrow \beta \text{, and conf} \end{cases}$

Minimum confidence is the least allowable confidence value of a rule that still allows it to be counted as an association rule. To count { A B } → { C } as a association rule, the minimum confidence should be less than 0.5 .

Association rules are generated by joining all the frequent sequences with other frequent sequences. There are some prerequisites to forming an association rule from two frequent sequences. Let (f_1, f_2) be frequent sequences to be join. The first prerequisite is that, the event count of sequence f_2 should be greater than the event count of the sequence f_1. The second prerequisite is that sequence f_2 start with sequence f_1. If the prerequisite conditions are met, then these two sequences make a rule. After that, a confidence check should be made and if the confidence of the rule is greater than the minimum confidence, then this rule is added to the association rules.

Frequent sequence and rule generation results should be stored in a permanent data storage unit, such as database or file system. In the thesis implementation, the frequent sequences are stored in the database table named tblFrequents, and the association rules are stored in the table tblRules. The diagram of the tables can be found in Appendix A.

3.2.2.7 Rule Pruning

Association rule pruning comes after generating all the association rules and is performed only if taxonomy exists on the ontology individuals. In this phase, uninteresting rules that are mentioned in the GSP algorithm [3] are pruned in order not to generate too many association rules.

An uninteresting rule is a rule whose support count is less than an expected value with respect to its taxonomical child. The expected value of any rule is the expected support count of the rule, relative to its taxonomical parent rule.

The taxonomical parent rule and child rule have the same number of items, and all items but one are equal and are at the same place in both sequences. The different item in a child rule is the taxonomical child of the different item in the parent rule.

The proportion of the support count of the different items in the parent rule to the support count of the different items in the child rule defines the expected proportion of the support count of the parent rule to the support count of the child rule. If the real support count proportion of the taxonomical parent rule to the taxonomical child rule is similar or less than the expected value, then the taxonomical child rule is uninteresting and is pruned from the association rules, since the support count of the child rule can be determined by the expected support count.

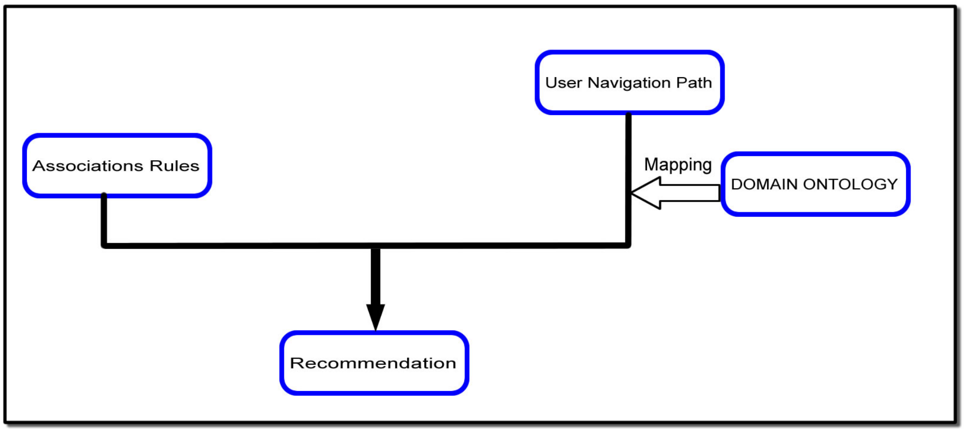

Figure 3.6: General Framework of Recommendation

3.2.3 Recommendation

3.2.3.1 Overview

Frequent sequence finding and rule generation are offline processes. These processes are routinely performed by a scheduled task when the workload of the server is at a minimum, and the results are stored in a permanent storage. However, the recommendation is an online process. As show in Figure 3.6 in the recommendation association rules and active user's nevigation history are used. In view of the user, an active navigation path to a new page or pages is recommended to the user in real time.

3.2.3.2 Window Count

Generally, not all the items in the active session path are taken into account while making a recommendation. A very early page that the user visited is less likely to affect the next page since users generally make the decision about what to click by the most recent pages. Therefore, the concept of window count is introduced. *Window Count* parameters define the maximum number of previous pages to be looked at while recommending a new page.

Since the association rules are in the form of ontology individuals, the user navigational history is converted into the sequence of ontology instances. Then the association rules and user navigation history are joined in order to produce recommendations.

In the recommendation phase, firstly the most recently navigated item is taken as a search pattern. All the association rules are scanned and the association rules whose antecedent part is equal to the search pattern are added to the *recommendation set*. This step continues with the last 2 items of the user's navigation path, and so on. This step iterates until the last window count item. If the window count equals 1, then the rule scanning is done only for the most recently navigated item.

Algorithm 3.2.8: RULEGEN($\mathcal{A}$, $window\_count, w < w_1, w_2, ..., w_n >$)

---

comment: Constructing recommendation set algorithm

comment: A is set of association rules w is active user navigation path

$T = \emptyset;$

for $i \leftarrow 1$ **to** $window\_count$

do for each rule R = $< r_1, r_2, ..., r_m > \Rightarrow < c_1, c_2, ..., c_j > \in \mathcal{A}$

do $\begin{cases} \textbf{if } r_m = w_n \text{ and } r_{m-1} = w_{n-1} \text{ and ... } r_{m-window\_count} = w_{n-window\_count} \\ \quad \textbf{then } T = T \cup R; \end{cases}$

The recommendation set is a set of association rules which will be used to make the recommendation. If there is no association rule in the recommendation set, then no recommendation will be made.

After constructing the recommendation set, the page recommendation starts. The association rules in the recommendation set are ordered by their confidence value and the highest one is taken first for the recommendation. For each association rule from the recommendation set, its consequent part is extracted. The consequent part of the rule contains ontology individuals; therefore, the instances should be converted to the real Web page. This conversion is done by examining the mapping ontology files created for the Semantic Web. For one ontology individual, there can be many Web pages to recommend. All the Web pages are recommended to the user. To avoid recommending too many pages to the user, a threshold value may be introduced.

This recommendation is made for every class of the domain ontology. For example, the SECKIN Web site server log has the concepts Authors, Vendor, PublishYear, and Category. For all classes, this recommendation phase should be run, and all the recommendation results are presented to the user.

3.2.4 Evaluation

3.2.4.1 Overview

The evaluation method of this work is based on the techniques introduced in [28]. The effectiveness of the recommendation is measured in both coverage and precision. Precision is the measurement of accurate recommendation percentages; on the other hand, coverage measures the ability of the recommender to produce all the items likely to be visited by the user.

Low precision means the recommendation engine makes inaccurate recommendations, and low coverage means the engine misses many recommendations that are likely to be visited by the active user.

In addition, two new evaluation metrics are introduced in the evaluation. These measurements are precision-with-threshold and coverage-with-threshold.

3.2.4.2 Experimental Dataset and Site Characteristics

For the evaluation, the server logs from 2 different actual Web sites are used. In addition, we have used a simple dataset from a dataset repository[39]. The first server log used for the experiments belongs to Seçkin Yayınevi (SECKIN) at the address www.seckin.com.tr and the other Web site log belongs to the department of Computer Engineering (CENG) of the Middle East Tecnical University at the address ceng.metu.edu.tr. The SECKIN Web site logs that have been used in this work are dated between 2008-01-20 and 2008-01-30 and the visit logs of the CENG Web site used in this work are dated between 2008-02-09 and 2008-03-09.

The SECKIN Web site is an online shopping site for books. This site allows the users to search for the books, view the book catalogs and view selected book information. There is a variety of book catalogs, such as catalog by category, newly added books, best-selling books and discounted books. The visitors can reach any book by viewing any of the catalogs or by using the search page.

There are about 10,000 books in 300 categories available to buy on the Web site. The site has approximately 190,000 page views daily. There remain 6,800 Web page view, and 1,700 unique sessions daily after the preprocessing of the server log. The average number of Web pages in a session is 4.

The ontology model of the domain includes the classes Authors, PublishYear, Category and Vendor. The Category class has a hierarchical structure, i.e. it includes taxonomy; the other classes do not include any taxonomy. There are 4545 Authors individuals, 249 Vendor individuals, 34 PublishYear individuals and 266 Category individuals.

As for the CENG site, it has many sub-websites in the CENG site. Some of them are the Web sites of individuals (i.e. students, teachers), news group (https://cow.ceng.metu.edu.tr/News/), and courses (https://cow.ceng.metu.edu.tr/Courses) . Most of the sub-websites other than the news group website, consist of only a few pages, and the daily visit count is quite small. Therefore, in the thesis, only the Web log of the CENG news group (http://cow.ceng.metu.edu.tr/) is used.

There are about 8,717 threads in 103 groups. The entire CENG site, including sub-Website has approximately 170,000 page views daily. After preprocessing, there are 17,000 page views and 1,100 distinct sessions daily. The average number of Web pages navigated is 15. In the news group ontology, classes of Authors, Thread and Group are presented. The individuals of Thread and Group classes are extracted from the server log. Since there is no clue for the Author of the thread, the instances of authors are randomly generated. It is thought that there are 500 students and faculty stuff in CENG; therefore, all threads are assigned to one of the 500 authors.

The other Web log is taken from UCI Knowledge Discovery in Databases Archive[39]. The dataset is the Web server log file of msnbc.com at http://kdd.ics.uci.edu/databases/msnbc/msnbc.html. This data contains the page visits of the users who visited msnbc.com on September 28, 1999. Visits are registered as one of the several pre-defined categories in time order. Only the category class of the Web site ontology is presented, and there are 17 distinct category individuals. There is no taxonomy on the instances.

3.2.4.3 Evaluation Methodology

A 10-fold cross-validation is performed for each 3 datasets. In each 10 iterations, the dataset is divided into two parts. The first part is the training part, taking 90% of the dataset, and the evaluation part takes 10% of the dataset.

Each transaction t in the evaluation set is divided into two parts. The first part is the first n items in t for recommendation generating. The other part, which is denoted as $eval_t$, is the remaining portion of t to evaluate the recommendation. N is the number of the window count.

After the recommendation engine produced a set of pageviews Rec_t , the set is compared with the remaining $eval_t$ pageview. Also, the presence of taxonomy is taken into account in the evaluation. If the recommended page is the taxonomical parent of the actual visited page, then the recommendation is evaluated as accurate; otherwise, it counts as inaccurate.

Precision is defined as the proportion of the number of relevant recommendations to the number of all recommendations. Precision measures the accuracy of the recommendation engine.

$$precision_t = \frac{|Rec_t \cap eval_t|}{|Rec_t|}$$

Table 3.8: Evaluation Parameters

s	Minimum support threshold
c	Minimum confidence threshold
n	Window Count
Rec_t	Generated recommendations for the transaction t.
$eval_t$	Part of the transaction to evaluate the recommendations
τ	presicion threshold (acceptence threshold)
σ	coverage threshold (acceptence threshold)

Coverage measures the ability of the recommendation engine to produce all the pageviews that are likely to be visited by the user. In other words, coverage measures how the recommendation covers all the pages that the user is likely to visit.

$$coverage_t = \frac{|Rec_t \cap eval_t|}{|eval_t|}$$

In the thesis, two new measurements are introduced, which are named "precision-with-threshold" and "coverage-with-threshold". The precision-with-threshold is an extension to the precision measurement. The value of new precision can be 0 or 1. If the precision is greater than the given precision threshold τ, then the value of precision-with-threshold is 1; otherwise, the value is 0.

$$precision\text{-}with\text{-}threshold_t = 0 \quad if \quad precision_t < \tau$$
$$precision\text{-}with\text{-}threshold_t = 1 \quad if \quad precision_t >= \tau$$

According to some, it is not very important to calculate the average value of all the precision values, because in their view, a recommendation is successful if the precision value is greater than a value in their mind; on the other hand, it is a total failure when the precision value is lower than the value in their mind. It is not very important for the precision to be a total success (i.e. 1) or to be the minimum threshold for them. For such circumstances, a precision-with-threshold measurement is introduced into the thesis. Precision-with-threshold evaluates the success of the recommendation precision value with respect to a given threshold value.

Moreover, precision-with-threshold helps to find the proportion of all the precision values to the given precision threshold value. Generally, if the precision-with-threshold values are lower than the precision values, then the precision values are generally lower than the precision threshold, and otherwise the precision values are generally higher than precision threshold.

Like the precision-with-threshold, the coverage-with-threshold is an extension to the coverage measurement. The value of coverage-with-threshold can be 0 or 1. If the coverage is greater than the given coverage threshold σ, then the value of coverage-with-threshold is 1; otherwise, the value is 0.

$$\begin{aligned} \textit{coverage-with-threshold}_t &= 0 \quad if \quad \textit{coverage}_t < \sigma \\ \textit{coverage-with-threshold}_t &= 1 \quad if \quad \textit{coverage}_t >= \sigma \end{aligned}$$

3.3 Implementation

Figure 3.7 depicts the general system design of the recommendation. The implementation is done using C# programming language. .Net framework library version 3.5 and the visual studio 2008 professional edition are used for implementation. As a permanent storage area, the Microsoft SQL Server 2005 developer edition is used.

3.3.1 System Decomposition

The system is developed in four packages.

LogParser : This package covers the classes that perform the Web server log parsing, session extraction and ontology mapping.

Shared : This package contains two classes which keep an item and taxonomy among the

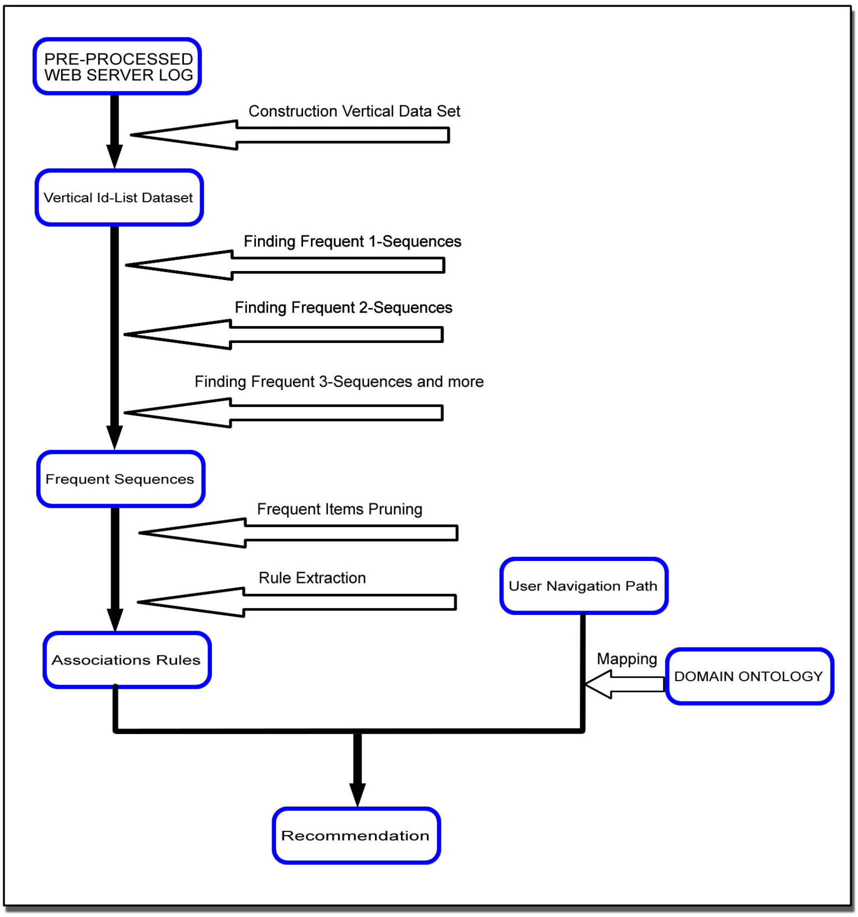

Figure 3.7: General Framework of Overall System Design

items.

SPADEX : This package covers classes that are specifically designed to implement the SPADE algorithm.

UI: This package contains classes related with the user interfaces.

3.3.1.1 Package LogParser

The classes of the Log File Parser package are shown in Figure 3.8. Log File Parser classes are responsible for pruning and parsing Web server log files as described in section 3.2.1.1.

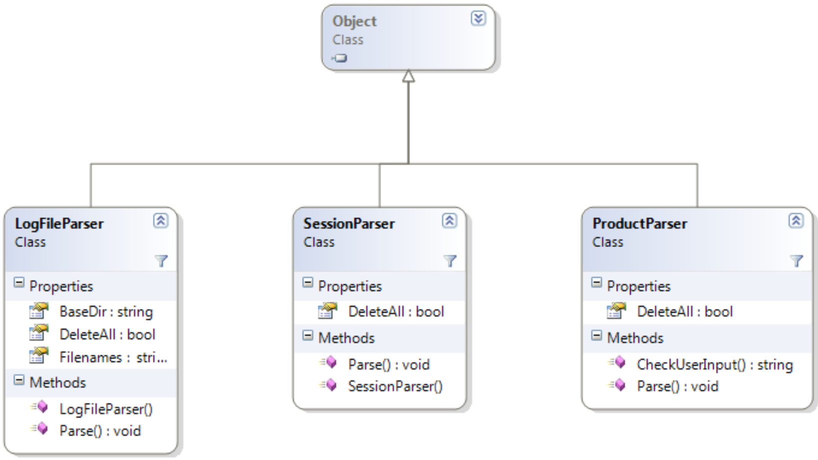

Figure 3.8: Classses in the the Logparser package

The session parser extracts the sessions from the pruned Web server log files. The results are stored in the tables tblSession and tblVisit. For the database diagram see Appendix A. The job of the product parser class is mapping between ontology instances and Web addresses as described in Section 3.2.1.3.

3.3.1.2 Package Shared

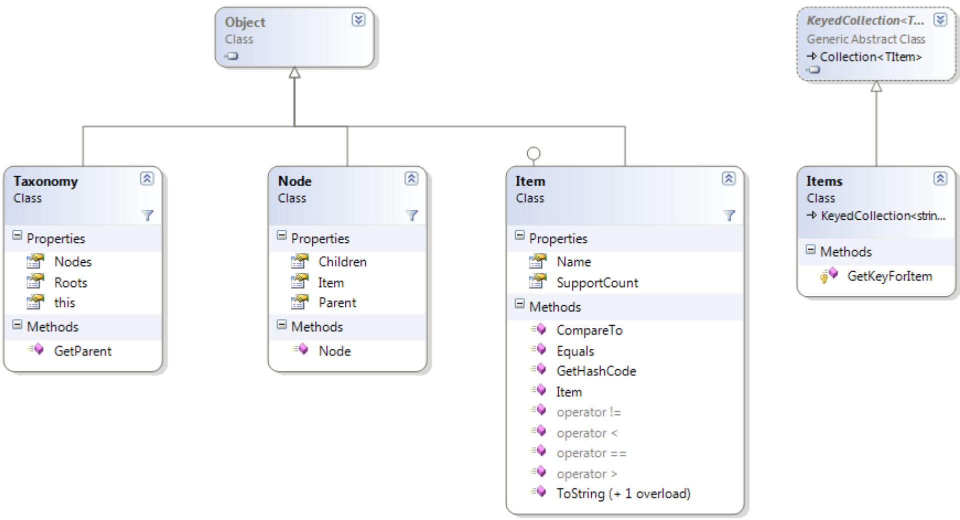

Figure 3.9: Classes in the Shared package

The shared package contains the commonly used classes in many different packages. The classes of the Shared package are shown in Figure 3.9. The most commonly used class is the Item class, which represents an item in a sequence. Usually the item is an individual of an ontology concept. The Items class holds a set of items. Taxonomy class keeps a hierarchical taxonomy of the items. The node class represents a node in the taxonomy.

3.3.1.3 Package SPADEX

The classes in the SPADEX package contain a set of classes that are related to the implementation of the SPADE algorithm. It is possible to divide the classes into two parts. The first set of classes is the data structure class that allows the representation of objects used in SPADE such as, sequence, event, etc. The other set is related to the SPADE tasks, such as finding 1-sequences, finding 2-sequences etc.

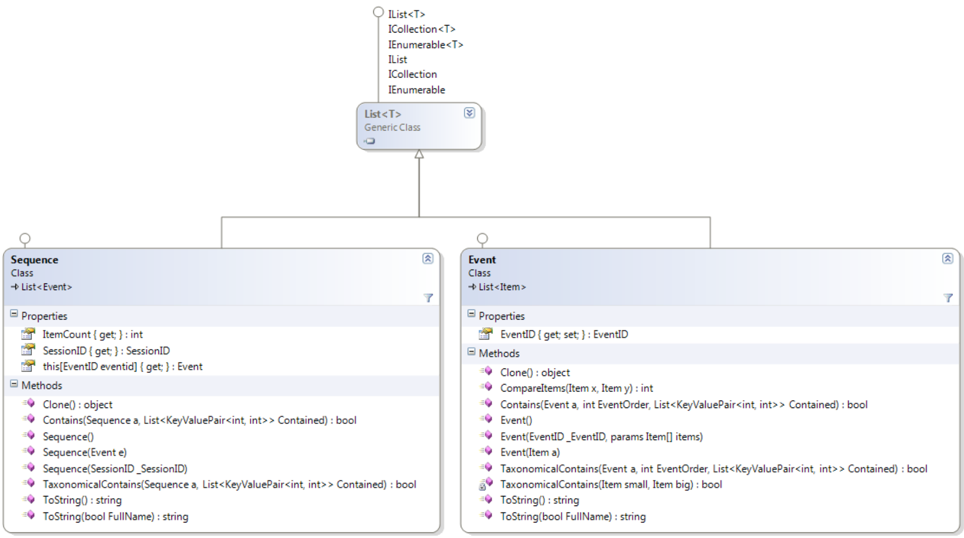

Figure 3.10: Data structure classes in the SPADEX package - 1

The Sequence and Event classes as shown in figure 3.10 are two backbone data structures of the SPADE algorithm. Sequence keeps an ordered set of events, and Event keeps an unordered set of items (without loss of generality, it is assumed that items are sorted in lexicographic order).The ”Contains” function of both classes tests whether a sequence contains another sequence, or whether an event contains another event.

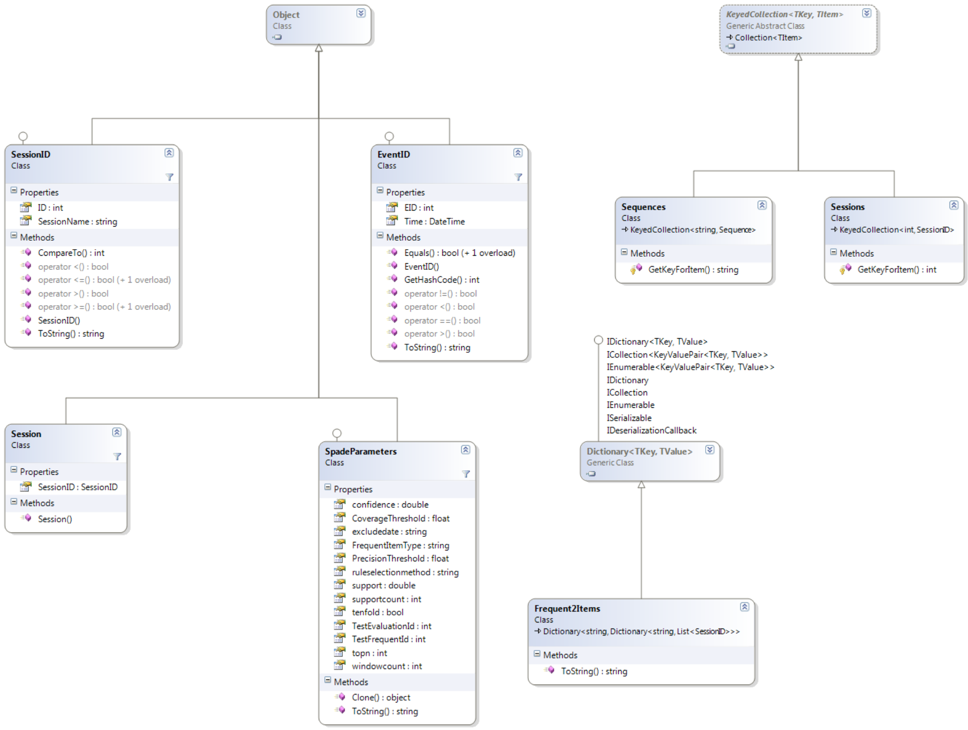

Figure 3.11: Data structure classes in the SPADEX package - 2

Figure 3.11 displays the remaining backbone data structure classes used for SPADE implementation. The Sequences class keeps a list of Sequence, and similarly the Sessions class keeps a list of sessions. The SessionID class is used to represent the ID of the session, and similarly EventID class represents the ID of an event. The Session class keeps meta-data of a session. Frequent2Items class is created to keep the 2-sequence candidates when finding the frequent 2-sequences (See section 3.2.2.3). The SPADEParameters class is designed to hold all the parameters of the SPADE algorithm. The confidence parameter holds the minimum confidence; support holds minimum support; PrecisionThreshold and CoverageThreshold keep the threshold for precision-with-threshold , and the threshold for new coverage-with-threshold respectively.

The Tenfold value is true if the dataset is divided into training and evaluation parts. If the Tenfold value is true, then the log of the day as given in the excludedate value is spared for the evaluation, and the other logs are used for training. The FrequentItemType value keeps the ontology concept that the algorithm runs. The WindowCount keeps the window count value for recommendation. The Ruleselection defines how many association rules are used for recommendation. The topn value is the percentage of the rules to be used for recommendation if the value of Ruleselection is topn.

The SPADEParameters class behaves like a bag of parameters. This class is used for SPADE and also many methods in SPADE take an instance of the SPADEParameters class.

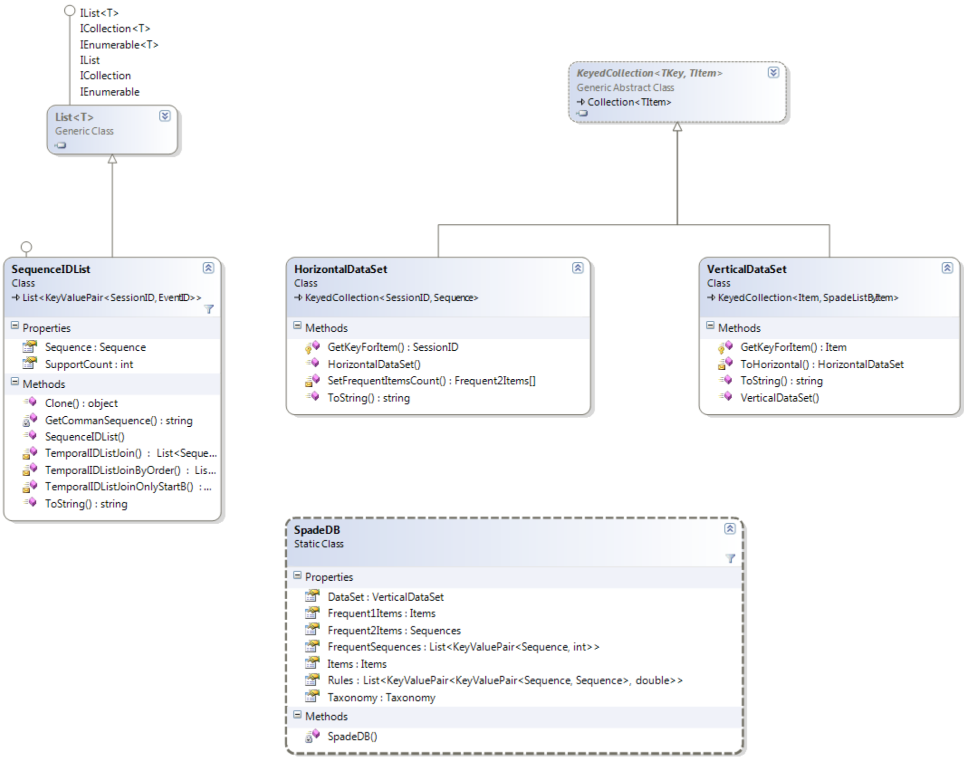

Figure 3.12: Complex data structure classes in the SPADEX package

Figure 3.12 depicts complex data structures for the SPADE implementation. The SequenceIDList class keeps the ID-list [7] of a sequence. The HorizontalDataSet class keeps a list of sequences along with their sequence IDs. The VerticalDataSet class represents a vertical layout transaction database. The VerticalDataSet class contains a set of items along with their ID-list values. SpadeDB is a static class that keeps the vertical dataset. Also, after the run, the frequent sequences are written to the properties Frequent1Items, Frequent2Items, and FrequentSequences of SPADEDB. The Rules property keeps the gathered association rules. Item property keeps all the distinct items in the dataset. Taxonomy keeps an object of the Taxonomy class and contains the taxonomy of the items.

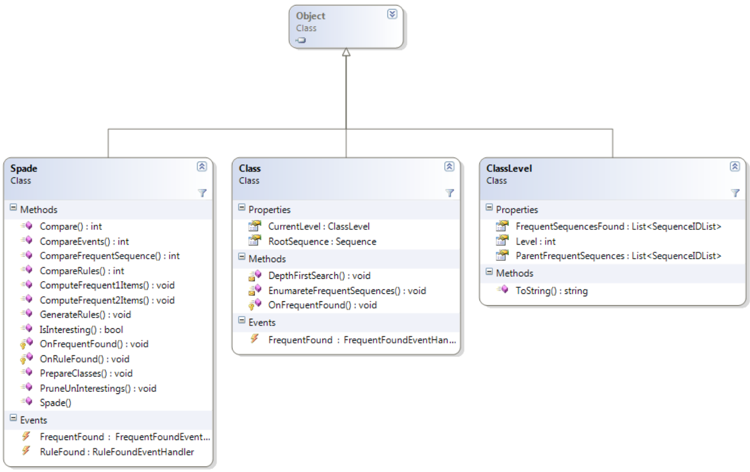

Figure 3.13: SPADE implementation classes in the SPADEX package

The classes depicted in Figure 3.13 perform actual SPADE algorithm. The SPADE class is the implementation of the SPADE algorithm. ComputeFrequent1Items and ComputeFrequent2Items methods find the frequent 1-sequnces and frequent 2-sequences respectively. The PrepareClasses method of SPADE class prepares the equivalence class for each item in the dataset, and starts iteration in the equivalence classes to find frequent sequences. The GenerateRules computes the association rules from the frequent sequences. The PruneUnInteresting methods prunes the uninteresting rules in cases of item taxonomy presence.

The Class and ClassLevel classes are auxiliary classes for the SPADE class. The Class represents an equivalence class. The ClassLevel represents each itereation on the equivalence class.

3.3.1.4 Package UI

The classes in UI packages are responsible for rendering user interface; also, there are some auxiliary classes that facilitate the communication between SPADE and the user interface.

Figure 3.14 depicts the auxiliary classes for the user interface. The Parser class is responsible for calling necessary methods from the classes in the LogParser package. The FrequentFinder

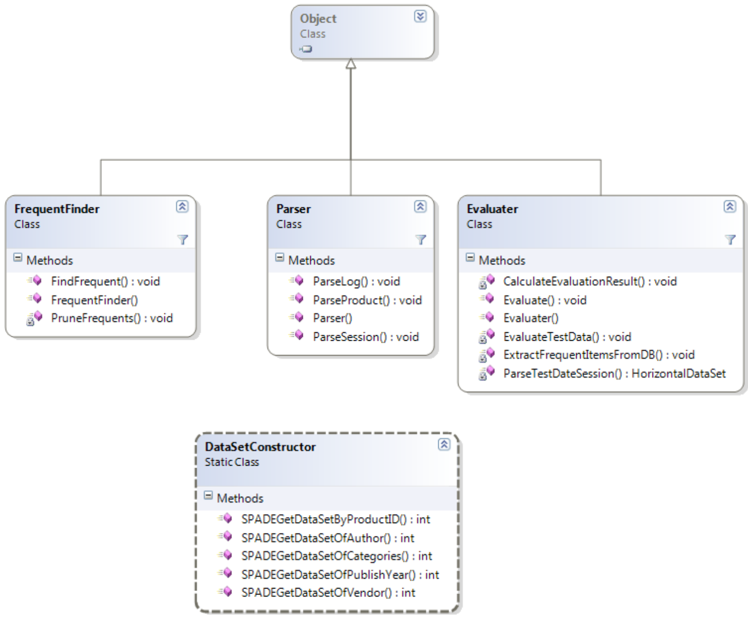

Figure 3.14: Classes in UI Package related with Interface SPADEX coordination

class divides the dataset into training and evaluation parts 10 times, and it runs the SPADE algorithm on the training dataset. The Evaluater class makes the recommendation for the evaluation part of the dataset for each 10 iterations. Also, it evaluates the result of the recommendation as described in 3.2.4. The DataSetConstructor class produces the vertical dataset from the stored data in the database.

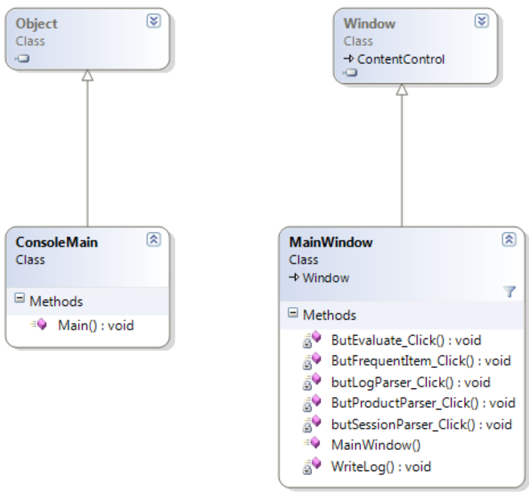

Figure 3.15: Classes in UI Package related with Interface

The classes responsible for rendering user interface are represented in Figure 3.15. There are two user interfaces. One is the console application used for batch jobs. The other class is the MainWindow class which produces the user interface as depicted in Figure 3.16. The class consists of click event methods for the button in the window form. Click event methods check the validity of the user inputs and call the necessary method from the auxiliary class shown in Figure 3.14.

The user interface of the system that is shown in Figure 3.16 is designed to allow the users to carry out all the steps from log parsing to evaluation. The program produces messages to display the progress of the task and the time taken by the task. These messages are shown in the Output section of the window form.

The ”Ontology Files” section allows the selection of the ontology files of the Web, which are OWL files. The ”Ontology Concept File” input is the entry point for the OWL file that contains the classes and relationships among classes. The ”Ontology Instance File” shows the ontology file that contains individuals of the class that are presented in the previous ontology file. The last ontology file the ”Ontology Mapping File” is the OWL file that keeps the mapping between web page addresses and ontology individuals.

There are three buttons in the parse logs section of the window. The first one is the ”Log Parser” button, which parses and prunes the Web site log files where a name is specified in the the ”Log File To Part” textbox above, and stores the result in the database. The second button is the ”Session Parser” that extracts sessions from the pruned visit log. The ”Ontology Mapper” button allows mapping between ontology individuals and Web page addresses. If the ”Delete Previous Logs” is checked, then the previous parsing result of the same configuration is purged from the database.

The ”Rule Finder” part of the user interface allows the finding of the frequent sequences and association rules. The support and confidence values can be changed by editing the appropriate textboxes. The ”Find Association Rules” button starts the process of finding frequent sequences.

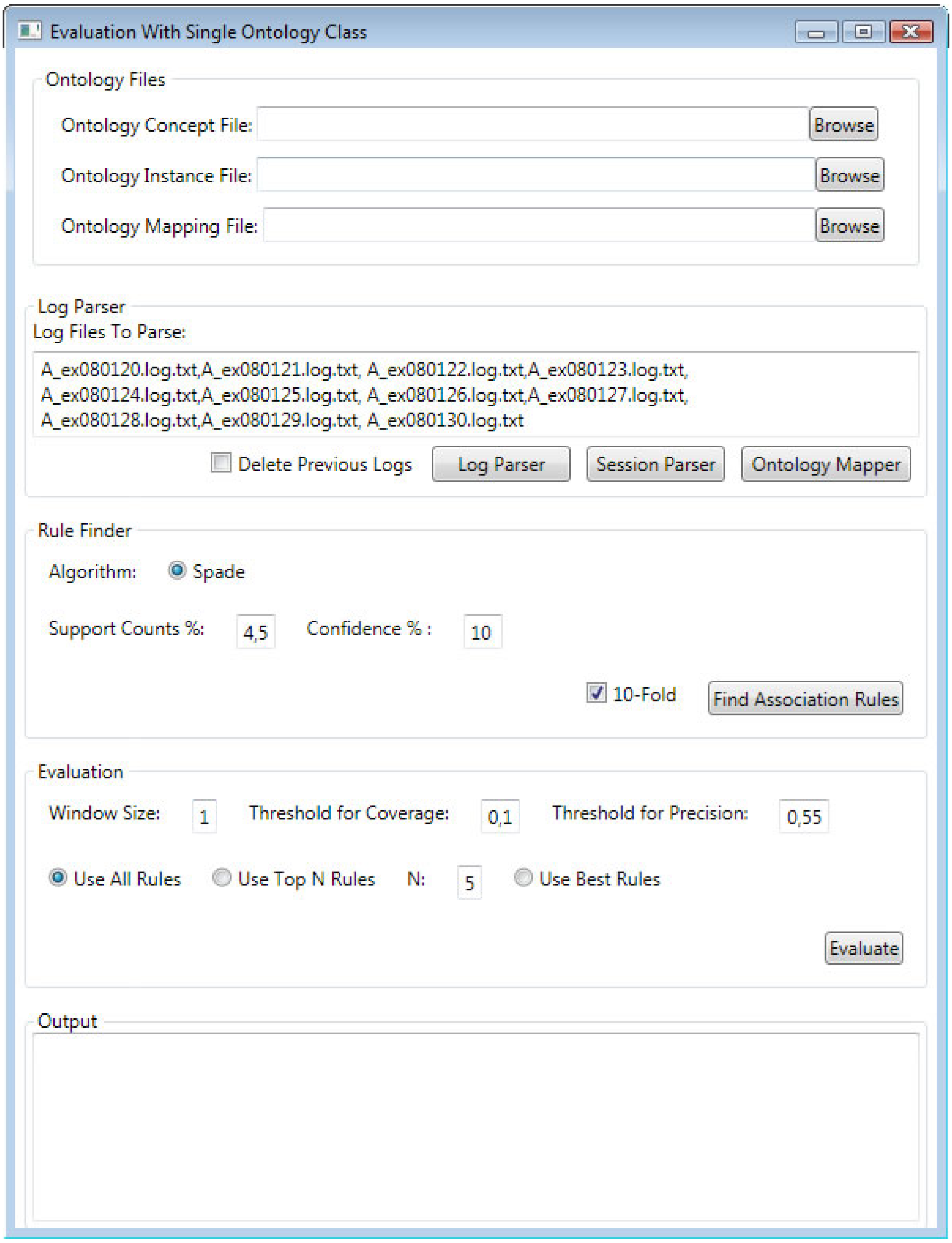

Figure 3.16: User Interface of the System

The Evaluation part of the user interface makes the recommendation and evaluates the result of the recommendation. The parameters (i.e. the window count, coverage threshold and precision threshold) can be changed by editing the appropriate textboxes. The radio buttons permit the selection of how many association rules will be used to make the recommendation. If ”Use All Rules” is selected, then all the association rules are taken into account when making a recommendation. When ”Use Top N Rules” is selected, only the top N percentage of the rules ordered by their confidence descendant is used to make recommendations. If the ”Use Best Rule” is selected, only the rule that has the maximum confidence value is used. The ”Evaluate” button starts the evaluation process.

CHAPTER 4

EVALUATION RESULTS

4.1 Overview

This section contains the results of the evaluations of the proposed system on three different datasets. For a given dataset and a set of parameters (i.e. minimum support threshold, minimum confidence threshold, window count, precision threshold value, coverage threshold value) a ten-fold cross-validation was performed as described in Section 3. In each 10 iterations, the dataset is divided into two parts. The first parts is the training part, taking 90% of the dataset, and the other part is evaluation, which takes 10% of the dataset.

For each session in the evaluation part of the dataset, recommendations are produced by joining the association rules and the navigation path under consideration in the session (i.e. the first window count events of the session). There can be zero, one or more recommendations for each session. The precision value for the generated recommendations of a session is calculated by dividing the number of accurate recommendations by the number of all recommendations. The coverage value for the generated recommendations is calculated by dividing the number of covered items in the session's evaluation part by the number of all items in it. Moreover the precision-with-threshold value for the generated recommendations of a session is assigned 1 if the precision value is greater than a pre-assigned precision threshold value; similarly, the coverage-with-threshold value for the generated recommendations of a session is assigned 1 if the coverage value is greater than the pre-assigned coverage threshold value.

The total precision and total coverage of an entire evaluation iteration is calculated by taking the average of all precision values and all coverage values of all sessions; similarly, the total new precision and total new coverage value of an entire evaluation iteration is calculated by taking the average of all new precision values and all new coverage values of all sessions.

These iterations are repeated 10 times. In each iteration, a different 10% portion of the data set is used for the evaluation, and the other portion is used for generating the association rules. The average of the total precision and coverage of all 10 iterations of an evaluation is calculated, and the result is the final precision and coverage values of the whole evaluation.

Four group of test are performed on the datasets to evaluate the recommendations. The first group of tests the effect of minimum support threshold to the precision and coverage values. The charts in second group of tests depict the effect of minimum confidence value to the precision and coverage values. The next group of tests investigates the effect of precision and coverage threshold values (acceptance threshold) to the precision-with-threshold and coverage-with-threshold values. In the last group of tests, the effect of number of association rules used in the recommendation to the precision and coverage values is revealed.

Certain conventions are used in the following sections. Generally, the words "association rules" are omitted and the words "'recommendation"' and "'evaluation"' are used alone. For example, "'the recommendations in terms of individuals of the Category class"' means, "'the recommendations that are generated from the association rules in terms of the individuals of the Category class"'. In addition, the word "recommendation" is omitted and only the word evaluation is used. For example, "'the evaluation in terms of individuals of the Category class"' means, "'the evaluation of the recommendations in terms of individuals of Category class"'. Moreover, the name of the ontology class implies all the individuals of the class. For example, "'recommendations in terms of Category"' means, "'recommendations that are generated from the association rules in terms of the individuals of Category class"'. The last convention is used in the captions of the evaluation charts. The captions of the figures consist of the three fields, which are: the class type of the association rules used to make recommendations, one or more values of the evaluation parameters, and the data set of the evaluation. For example the caption "Book, window count = 1, SECKIN Web Site" means, in the evaluation, the recommendations are generated from the association rules in terms of individuals of the Book class, the window count value is 1, and the log file and ontology of the SECKIN web site are used.

4.2 Evaluation of the effect of minimum support threshold

This section shows the results of evaluation with respect to different minimum support thresholds. Increasing the minimum support threshold value, would limit the number of the generated frequent sequences, and would also limit the number of recommendations generated. The behavior of the precision and coverage values in case of changing the minimum support threshold is depicted in the following charts.

In the first part the effect of the minimum support threshold to the recommendation for each class is depicted, and the next part shows the effect of minimum support threshold to the recommendation when the recommendation is made instead of assiciation rules of a single class, it is made by combining all association rules of all classes.

4.2.1 Effect of minimum support threshold using single class rule

Different association rules are produced using different classes in web usage mining; this would lead to different recommendations for each different class. In this section the precision and coverage values of these recommendations are depicted using the single class rule.

The SECKIN data set has five classes, the MSNBC dataset has one class while CENG dataset has three classes, so distinct evaluations are made for all nine classes.

The following constant parameters are used for the evaluations on the SECKIN data set.

Data Set: SECKIN Web Site
Minimum Confidence: 10%
Threshold Precision:0.3
Threshold Coverege:0.3

Rule Selection: All Rules

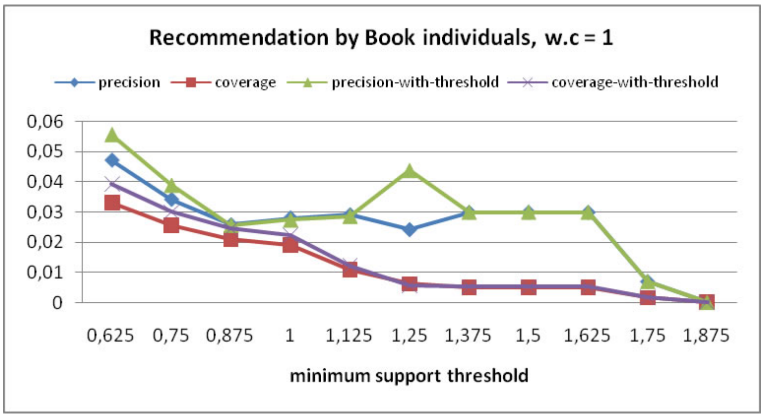

Figure 4.1: Book class, window count = 1, SECKIN Web Site

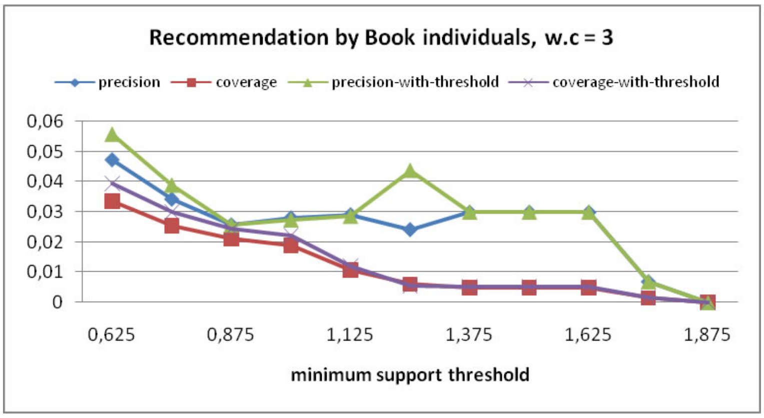

Figure 4.2: Book class, window count = 3, SECKIN Web Site

Figures 4.1 and 4.2 display the evaluation results in terms of the Book class and their window count parameters are 1 and 3 respectively. An individual of the Book class holds all the information for a single book in the SECKIN web site. For each book on the website a single dedicated web page is generated; hence, a single web address corresponds to a single book individual. Therefore the navigation path in terms of the Book class is parallel to the navigation path in terms of web address. Therefore, recommendation in terms of the Book class is similar to the recommendation in terms of web page addresses.

The number of distinct Book individuals is 6,387. Since this number is relatively large, to get an association rule, the minimum support threshold should be very small and generated association rules do not depict the general preference of the visitors, since the support ratio of the rule is generally very small. Therefore, the precision and coverage values of the evaluation are small. As expected, when the minimum support increases, the precision and coverage values decrease, since the number of association rules decreases. The window count parameter does not change the evaluation result, and this shows that generally the recommendation is made only using the last item of the user navigation history and furthermore that the antecedent parts of the association rules are association containing only a single item.

The time of the evaluation becomes extremely long when the minimum support count gets smaller, because the number of frequent sequences increases exponentially when the minimum support threshold decreases. Therefore, the time taken to find association rules to make recommendation for a session increases exponentially as well. Therefore for the above test, it is not feasible to make a test with a minimum support threshold less than 0.625%.

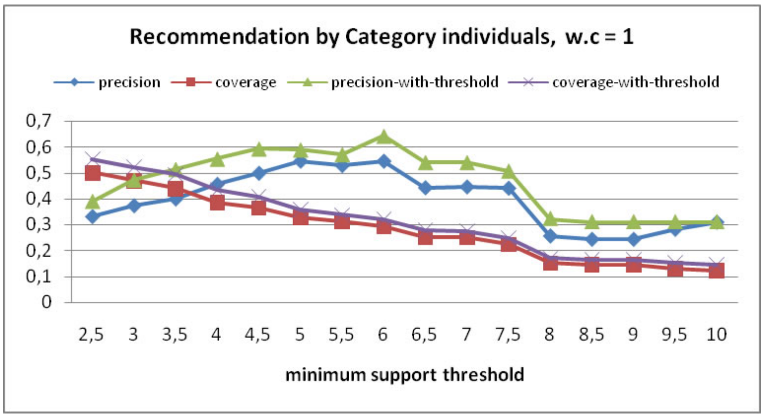

Figure 4.3: Category class, window count = 1, SECKIN Web Site

Figures 4.3 and 4.4 display the the evaluation results in terms of the Category class and their window count parameters are 1 and 3 respectively. The number of distinct Category individuals is 286, and as distinct from the other classes, the Category individuals have a taxonomy relation. There are 20 root individuals, and 198 direct children of these root individuals in addition to 68 individuals that are grandchildren of the root individuals.

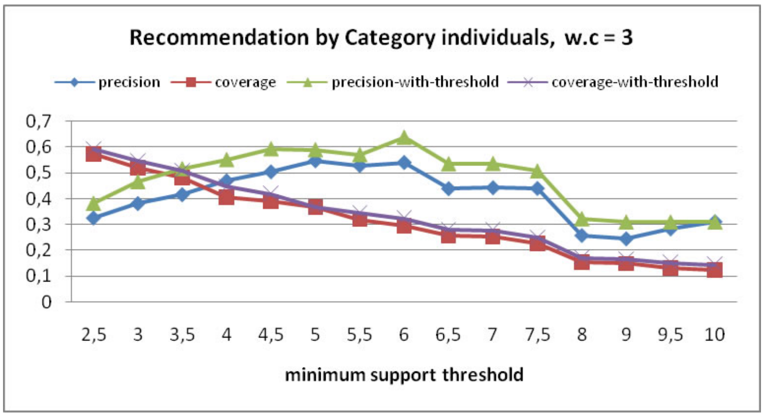

Figure 4.4: Category class, window count = 3, SECKIN Web Site

Since the number of individuals of the Category class is smaller than those of the Book and Author classes, the generated association rules have higher confidence and support count. Therefore, the precision and coverage values of the evaluation are relatively higher than in the aforementioned results.

At first, the precision value increases when the minimum support increases. The reason for this is that when the minimum support increases, the number of association rules decreases and the proportion of number of the accurate association rules to the number of all rules decreases because irrelevant association rules are eliminated. After a breaking point (6% minimum support threshold) the precision value starts to decrease, because the number of association rules decreases. In fact some sessions cannot get any recommendations at all and their precision value assigned is 0. When the number of sessions whose precision is 0 increases, then the overall average of the entire session's precision decreases.

Coverage value decreases constantly when minimum support increases. This is expected since the coverage is the proportion of covered items in the session's evaluation part to the number of all items in that part, and the number of items in the session's evaluation part is always constant, but the number of covered items in the session's evaluation part decreases when the minimum support count increases. Therefore, the coverage value always decreases.

At a small minimum support threshold value, increasing the window count also increases the precision and coverage values little, but after a threshold value (minimum support count =

5.5%), the window count does not affect precision and coverage values.

The results of the evaluations in terms of other classes of SECKIN web site can be found in Appendix E.

The following charts contain the results of the evaluation for the MSNBC web site log files in cases of different minimum support threshold value. The Ontology of the MSNBC web site contains only a single class, which is Category, therefore the evaluation is done for just that single class.

The following constant parameters are used for the evaluations on the MSNBC dataset.

Data Set: MSNBC Web Site
Minimum Confidence: 10%
Threshold Precision: 0.3
Threshold Coverege: 0.3
Rule Selection: All Rules

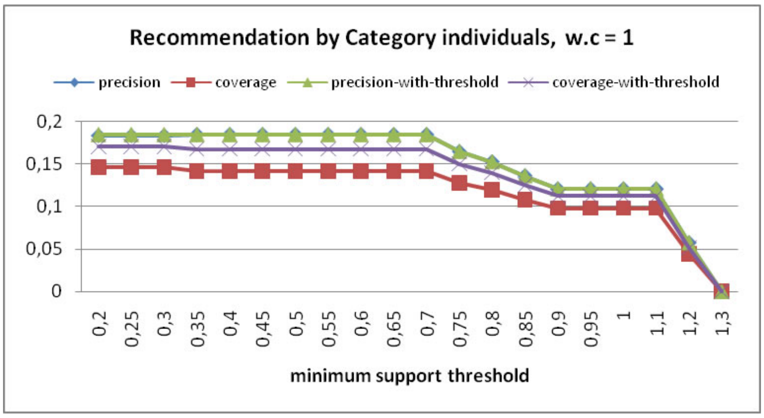

Figure 4.5: Category class, window count = 1, MSNBC Web Site

Figures 4.5 and 4.6 display the evaluation results in terms of the Category class and their window count parameters are 1 and 3 respectively. The number of distinct Category individuals is 17. The precision and coverage value after minimum support threshold 1,3 is 0 because after

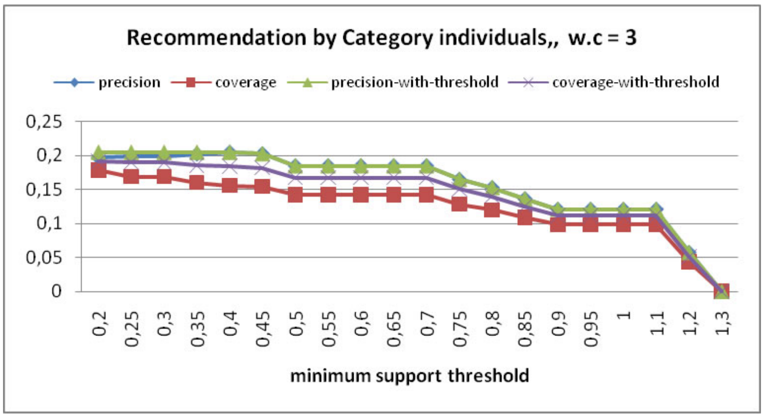

Figure 4.6: Category class, window count = 3, MSNBC Web Site

that minimum support threshold value, no association rules are produced. The precision and coverage values constantly decrease between the minimum support threshold values 0,7 and 1,3. At these minimum support threshold values, the number of generated association rules is decreasing and this reduces the value of precision and coverage. One characteristic of the data set is that the number of association rules does not change when minimum support count is less than 0,7.

The evaluations show some similarity to the SECKIN data set evaluation. In both evaluations the coverage values decrease. However in the SECKIN web site evaluation, the precision values increase prior to decreasing. In the MSNBC data set, precision value only decreases. In fact, at low minimum support, the precision value also increases, but due to time limitations, the evaluation is not feasible below a minimum support threshold 0.2%.

The following charts display the evaluation results of the CENG web site. The ontology of the CENG web site contains 3 classes which are Author, Group and Thread. Evaluation in terms of different minimum support thresholds are carried out for each of the three classes.

The following constant parameters are used for the evaluations on the CENG dataset.

Data Set: CENG Web Site

Minimum Confidence: 10%

Threshold Precision:0.3

Threshold Coverege:0.3

Rule Selection: All Rules

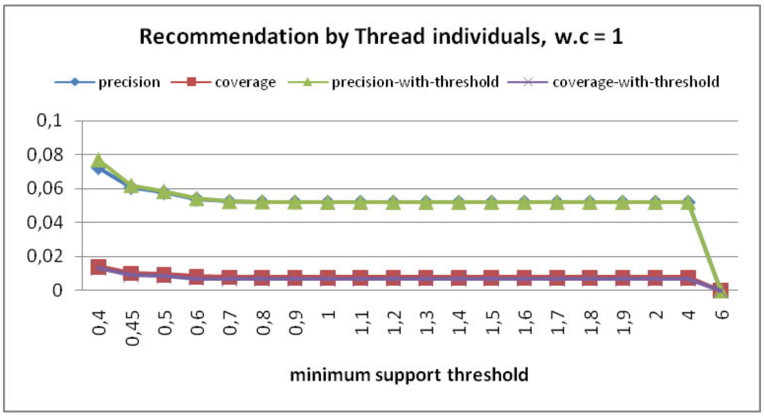

Figure 4.7: Thread class, window count = 1, CENG Web Site

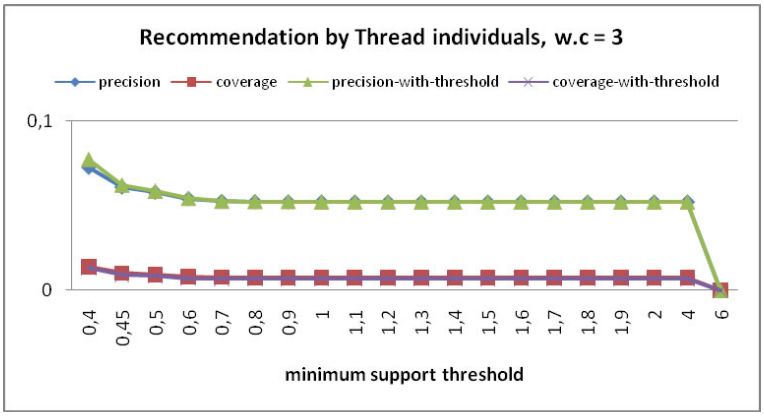

Figure 4.8: Thread class, window count = 3, CENG Web Site

Figures 4.7, and 4.8 shows the evaluation results in terms of the Thread class and their window count parameters are 1 and 3 respectively. The number of distinct Thread individuals is 8,717. Since the number of items is quite large, association rules are extracted at a small minimum support percentage. Therefore, minimum support starts with 0.4% . Below 0.4%, the evaluation cannot be executed in a feasible time. Like the MSNBC dataset, the number of association rules does not change significantly when the minimum support changes. Therefore, the precision and coverage values do not change between minimum support 0.5% and 4%.

After minimum support 6%, no association rule is produced, so the precision and coverage values are 0.

The window count parameter changes neither the precision nor coverage values. Since the number of association rules in terms of the Thread class is very low, the window count parameters do not contribute any recommendations.

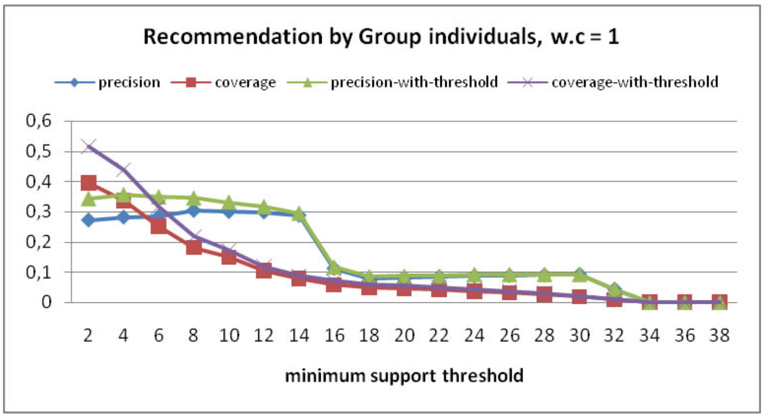

Figure 4.9: Group class, window count = 1, CENG Web Site

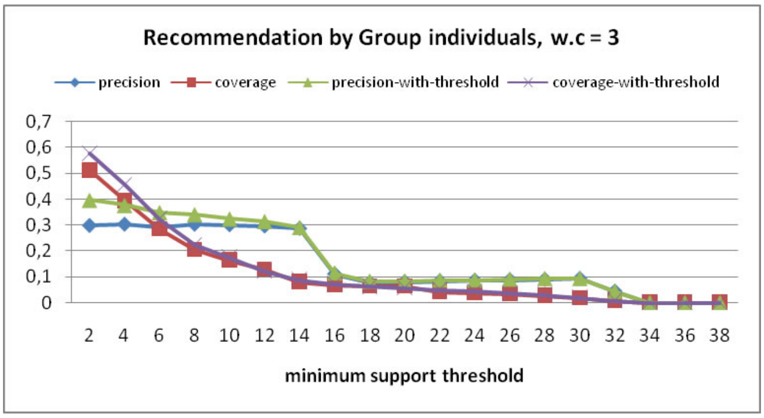

Figure 4.10: Group class, window count = 3, CENG Web Site

Figures 4.9 and 4.10 shows the evaluation results in terms of the Group class and their window count parameters are 1 and 3 respectively. Evaluation of other classes can be found in Appendix F. The number of distinct Group individuals is 103. The coverage value shows smooth decrease similar to that in the evaluation on the SECKIN web site. As with the SECKIN web site log, at first the precision value increases until a breakpoint value, (i.e. minimum support 6%). The precision value shows a sharp decrease at minimum support 14, but at other values the precision value decreases very smoothly.

Since the number of association rules in terms of the Group class is higher than that is in Author class, the window count parameter changes the precision and coverage values. Increasing the window count parameter increases the precision value as well conversely, it decreases the coverage value.

All charts show that, when the number of distinct individuals is small, then the more accurate recommendations are produced (high precision value) and the recommendation covers more items that the user is likely to visit (high coverage value). When the minimum support threshold increases, the precision value starts to increase until a breaking point, and after that breaking point the precision values start to decrease. The coverage value always decreases when the minimum support threshold value increases. After a certain minimum threshold value, the precision and coverage values drop to zero, since no association rules can be produced after that minimum support threshold.

The window count parameter does not generally change the precision and coverage values or changes them only slightly. This shows that the recommendations are generally made using only the last item of the recommendation part of the session, and the recommendations that use more than one of the last items are very rare.

4.2.2 Effect of minimum support threshold using all rules

This section contains the evaluation of recommendations made by combining all the individuals. The recommendation is made for each class. Zero, one or more recommendations are produced for each session in the evaluation data set. The precision, coverage, new precision, and new coverage values are calculated for each session after that the evaluation results are combined. Combination is made by taking the maximum values of each session from each class. That means a combined precision value is the maximum precision value among the recommendations for the session in terms of different classes; similarly, coverage value is calculated by taking the maximum of all coverage values for a session.

However, the combination of each class by equal numbers of minimum support is not feasible, since in each class, below some breaking point, the association rules cannot be extracted in a timely manner. This breaking point for the Book class is 0.625 and for PublishYear class is 5. Therefore, at equal minimum points, the classes cannot be combined. So, a proportion between the minimum support thresholds is used. This equation is computed by using the breaking point of each class.

$$\delta(Author) = \delta(Category)\ /\ 2 = 2\ x\ \delta(Book) = \delta(PublishYear)\ /\ 4 = \delta(Vendor)\ /\ 2$$

The following parameters are used for each evaluation.

Minimum Confidence: 10%

Threshold Precision:0.3

Threshold Coverege:0.3

Rule Selection: All Rules

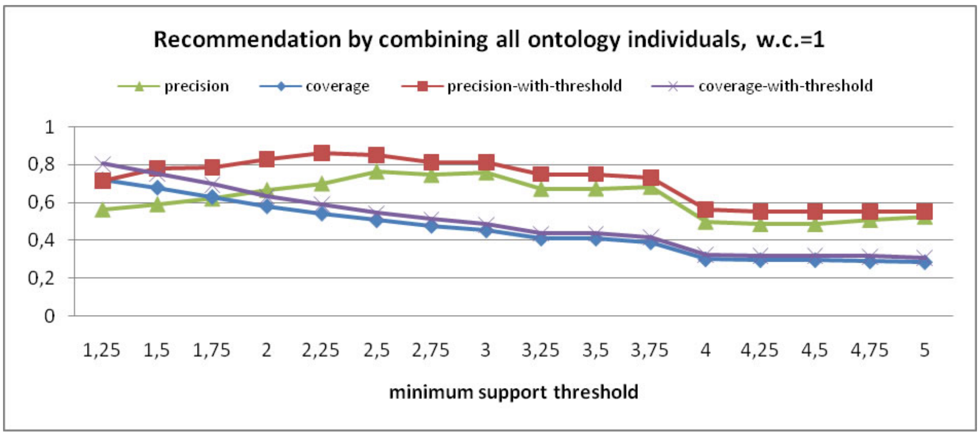

Figure 4.11: All classes, window count = 1, SECKIN Web Site

Figures 4.11 and 4.12 display the evaluation result in terms of the combination of all classes

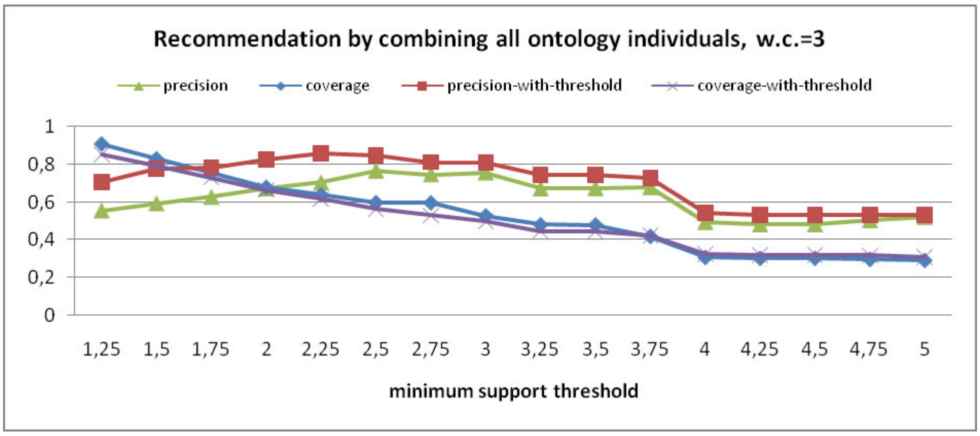

Figure 4.12: All classes, window count = 3, SECKIN Web Site

in the SECKIN Web site, and their window count parameters are 1 and 3 respectively.

At first, the precision value increases when the minimum support count increases. This is expected behavior since all the precision values of the evaluations in terms of the single classes first increase until there is a breaking point. After the breaking point (2.5%) the precision values start to decrease.

The coverage value decreases constantly when the minimum support increases. This is expected since all the coverage values of the evaluation in terms of single classes decrease constantly.

At the small minimum support, increasing the window count also increases precision and coverage values a little, and after a threshold value (minimum support count = 5.5%) the window count does not affect the precision and coverage values.

Since the MSNBC web site has only one class, the result of the evaluation using all rules is the same as the result of the evaluation using only the Category class.

The following part contains the evaluation result of recommendation by combining all classes in the CENG web site. Similar to the SECKIN web site, a proportion between the minimum support thresholds is used. This equation is computed by using the breaking point of each class.

$$\delta(Group) = 4\ x\ \delta(Thread) = 4\ x\ \delta(Author)$$

The following parameters are used for each evaluations.

Minimum Confidence: 10%

Threshold Precision:0.3

Threshold Coverege:0.3

Rule Selection: All Rules

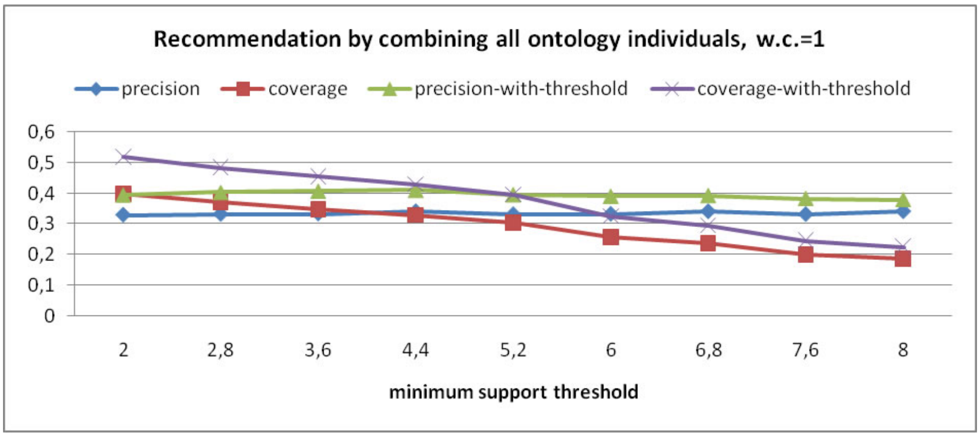

Figure 4.13: All classes ,window count = 1, CENG Web Site

Figures 4.13 and 4.14 display the evaluation results in terms of the combination of all classes in the CENG Web site, and their window count parameters are 1 and 3 respectively.

The precision value shows similar behavior to that of the Group class. This is expected since the maximum precision values generally belong to the Group class. The precision value increases when the minimum support increases.

The coverage value decreases constantly when the minimum support increases. This is ex-

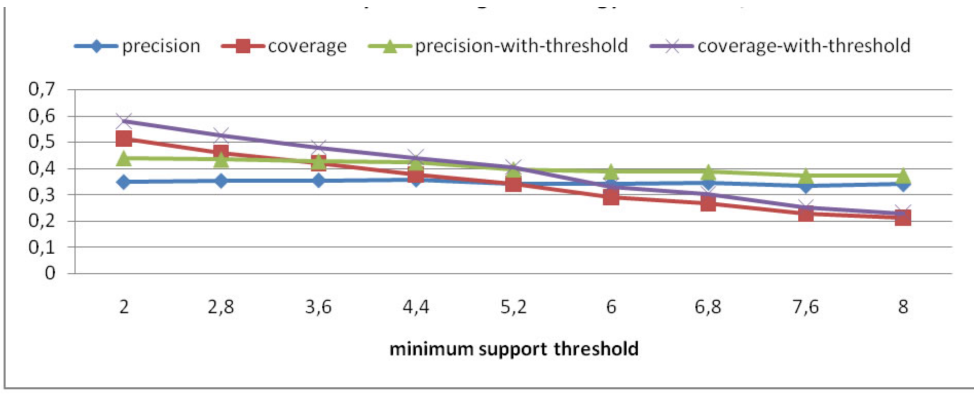

Figure 4.14: all classes ,window count = 3, CENG Web Site

pected since all coverage values of the evaluation in terms of single classes decrease constantly.

All charts depict that, combining all association rules generates more accurate recommendations than those generated using association rules of a single class. Also, the coverage value is higher than the coverage value of each class.

Since the window count parameter makes a slight change to the precision and coverage values, in recommendation with combined rules, the window count parameter makes slight changes in the precision and coverage values.

4.3 Evaluation of the effect of minimum confidence threshold

This section shows the results of evaluations in cases of different minimum confidence thresholds. Increasing the minimum confidence threshold value would eliminate many less confident rules and would decrease the number of generated association rules, it could also would limit the number of generated recommendations. The behavior of the precision and coverage values in cases of changing the minimum support threshold is depicted in the following charts.

4.3.1 Effect of minimum confidence threshold using single class rule

In this section, the precision and coverage values of the recommendations are depicted using only a single class rule.

The following parameters are used for each evaluation in the SECKIN web site.

Threshold Precision:0.3

Threshold Coverege:0.3

Rule Selection: All Rules

Window Count:1

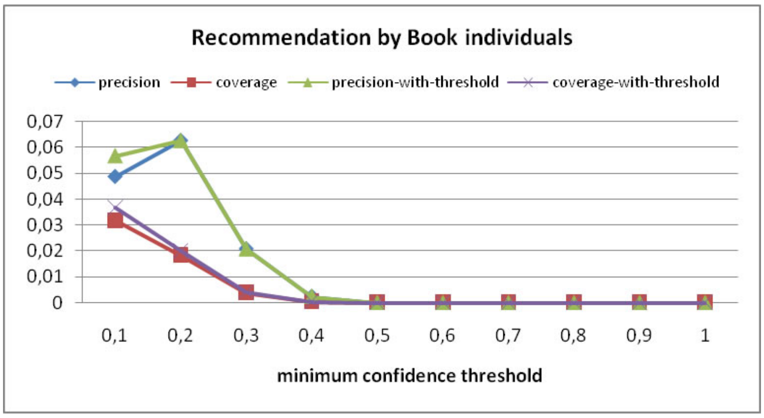

Figure 4.15: Book class, Minimum Support Threshold: 0.675% , SECKIN Web Site

Figures 4.15, 4.16 display the evaluation results in terms of the Book and Category class respectively. Evaluation of other classes can be found in Appendix G.

As with dynamic minimum support evaluation, with dynamic minimum confidence, the precision values increases until a breaking point (breaking points are generally 0.2 and 0.3) after which, the precision starts to decrease. The reason for this is that the number of association rules decreases when minimum confidence increases. Therefore, a number of irrelevant rules begins to be eliminated and the proportion of the accurate association rules increases. Af-

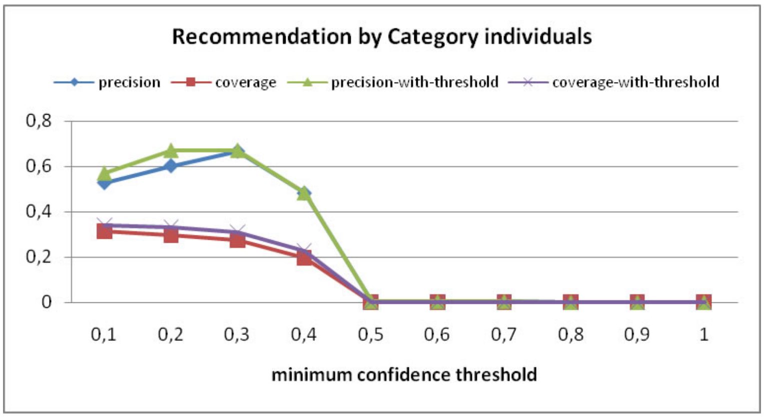

Figure 4.16: Category class, Minimum Support Threshold: 5.5% , SECKIN Web Site

ter the breaking point, the number of association rules becomes very low and the number of sessions whose recommendation count is 0 increases. Therefore, many sessions get 0 as the precision value, and the overall precision value starts to decreases.

The coverage value decreases steadily when the minimum confidence decreases, because when the number of association rules decreases, then the number of covered items in each session's evaluation part decreases as well, and this makes the coverage value smaller.

After a breaking point, both precision and coverage values reach zero. That means after this point there is no association rule that can be used for recommendation.

The following parameters are used in the next evaluation which used the MSNBC data set.
Window Count:3
Threshold Precision:0.3
Threshold Coverege:0.3

Figure 4.17 displays the effect of minimum confidence threshold on the precision and coverage values.

Unlike previous evaluations, the precision values do not increase until a breaking point, they

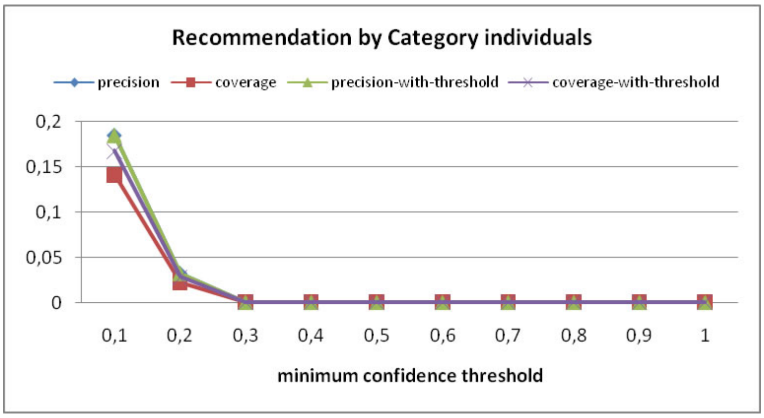

Figure 4.17: Category class, Minimum Support Threshold: 0.6% , CENG Web Site

start decreasing from the first minimum threshold value. The reason for this is that the confidences of the association rules are small; therefore, increasing the minimum confidence value by a small amount, drops the precision value dramatically.

The coverage value decreases steadily when the minimum confidence decreases. because when the number of association rules decreases, then the number of covered items in each session's evaluation part decreases as well, and this makes the coverage value smaller.

The following parameters are used in the next evaluation, which used the CENG data set.

Window Count:1
Threshold Precision:0.3
Threshold Coverege:0.3

Figure 4.18, display the evaluation results in terms of the Group class. Evalution of other classes can be found in Appendix H.

The Author and Thread classes show similar behavior to the Category class of the MSNBC dataset, since the number of generated association rules is small in terms of these classes.

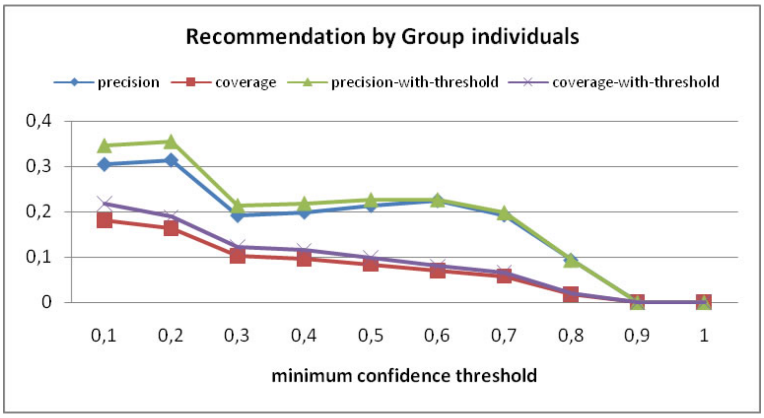

Figure 4.18: Group class, Minimum Support Threshold: 8% , CENG Web Site

Since the number of association rules is relatively high and their confidence is relatively high, the precision and coverage value does not drop to zero immediately in the chart of the Group class. Confidence values of some association rules are as high as 85%, so, at a high minimum confidence threshold, precision values exist.

4.3.2 Effect of minimum confidence threshold using all rules

This section contains the evaluation of recommendations by combining all classes. As described in Section 4.2.2, the precision, coverage, precision-with-threshold and coverage-with-threshold values are calculated for each session; after that, the evaluation results are combined. Combination is done by taking the maximum values of each session from each class. The same ratio of minimum support threshold between ontology classes are used as described in Section 4.2.2.

Figure 4.19 shows the effect of minimum threshold value on the precision and coverage values when the recommendation is done by combining the rules of all classes.

Similar to the evaluation of single classes, the precision value increases until a breaking point, whose value is the maximum of the breaking points of the evaluation of single classes which is 0.3. After that breaking point, the precision value starts to decrease since the number of

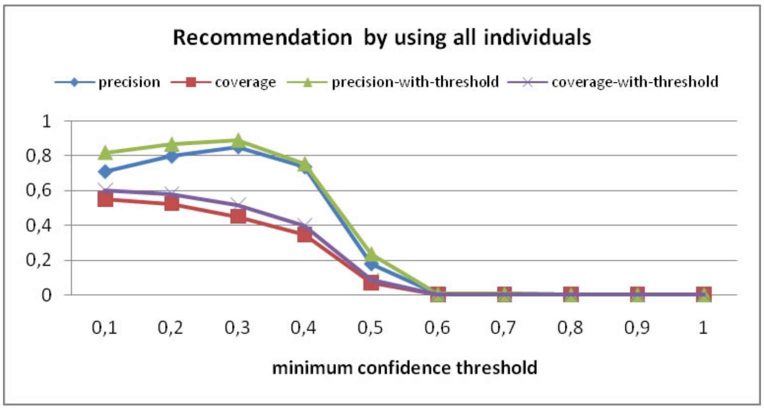

Figure 4.19: All classes, SECKIN Web Site

combined association rules decreases.

The coverage value decreases when the minimum confidence threshold value increases, since when the minimum confidence threshold gets higher, the number of combined association rules decreases; hence, the number of items of the session covered by the recommendations decreases.

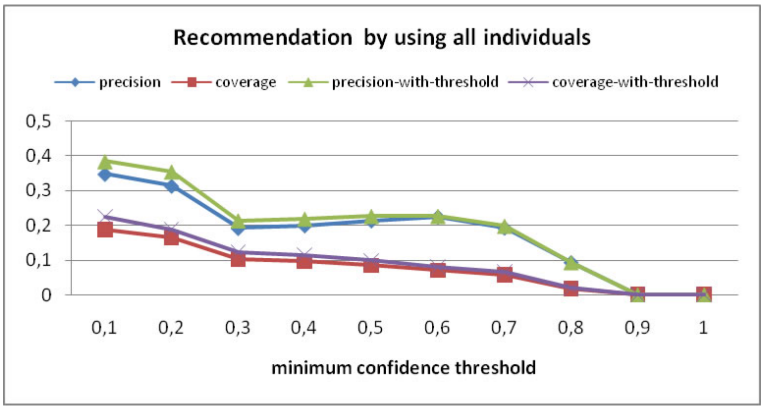

Figure 4.20: All classes, in CENG Web Site

Figure 4.20 shows the effect of the minimum threshold value in cases of combined association rules in the CENG web site.

The Group class dominates the overall test result, since, as shown in the previous test results, the other classes (Author and Thread) produced a very low number of association rules. Therefore, mostly the precision and coverage values of the Group class are presented in the chart, although all the association rules are combined.

4.4 Evaluation of the effect of precision and coverage thresholds

Changing the precision and coverage threshold values changes the precision-with-threshold and coverage-with-threshold values respectively. The precision and coverage threshold values behave like acceptance points and measure the success of the precision and coverage values with respect to a given threshold value. In this evaluation, the effect of changing precision and coverage thresholds on the precision-with-threshold and coverage-with-threshold values is revealed.

4.4.1 Effect of precision threshold

This section shows the result of evaluation in terms of different precision threshold values. For each ontology class, the recommendation is run separately. The following parameters are used for each evaluation.

Minimum Confidence :10% Threshold Coverege:0.3
Rule Selection: All Rules
Window Count:1

Figures 4.21, 4.22 display the evaluation results in terms of the Book class, Category class respectively. Evalution in terms of other classes of SECKIN Web site can be found in Appendix I.

When the precision threshold value is 0, then the value of precision-with-threshold is 1. Be-

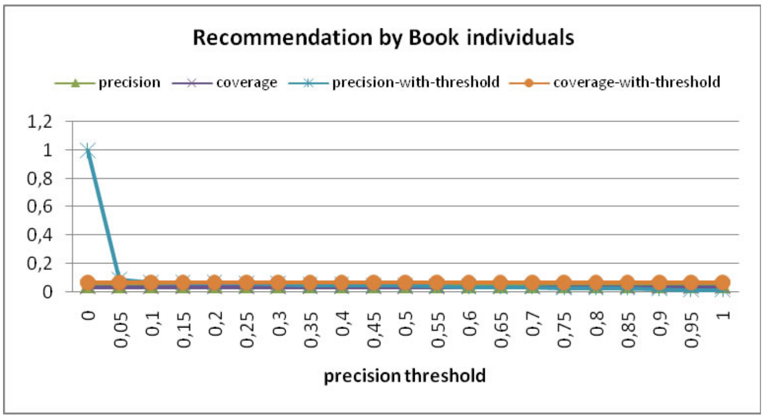

Figure 4.21: Book class, Minimum Support Threshold: 0.575% , SECKIN Web Site

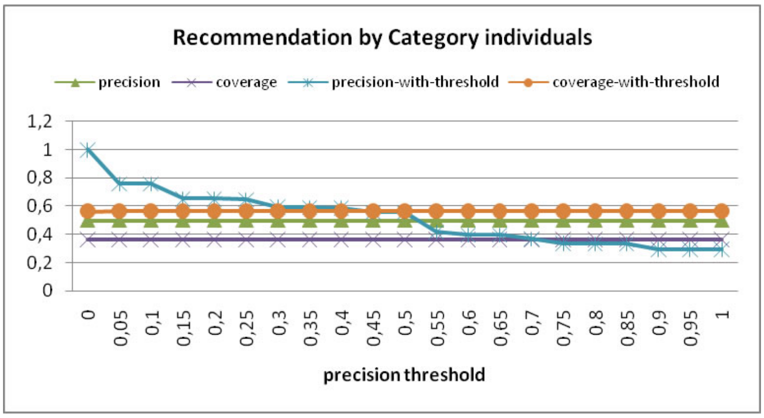

Figure 4.22: Category class, Minimum Support Threshold: 4.5% , SECKIN Web Site

cause all the precision-with-threshold values of the each session will be 1 even though their precision values are 0 when the precision threshold value starts to increase, then the value of precision-with-threshold starts to drop. If the average precision value is far from 1, then a sharp decrease occurs (as in the case of Author and Book); otherwise, a smooth decrease occurs (as in the case of PublishYear, Vendor and Category). When the precision threshold reaches 1, then the value of precision-with-threshold reaches its minimum.

In Appendix J effect of precision threshold in terms of classes of CENG web site is presented. When the precision threshold value is 0, then the value of precision-with-threshold is 1. Since the precision of the Author and Thread classes is very small, a sharp drop occurs in the precision-with-threshold values of these classes. On the other hand, since the precision value of the recommendation in the Group class is relatively higher, a steady decrease occurs in this evaluation.

4.4.2 Effect of coverage threshold

This section shows the results of evaluations in terms of different coverage threshold values. For each ontology class, the recommendation is run separately for each class. The following parameters are used for each evaluation.

Minimum Confidence :10% Threshold Precision:0.3
Rule Selection: All Rules
Window Count:1

Figures 4.23 and 4.24 display evaluation results in terms of the Book class, Category class respectively. Evalution in terms of other classes of SECKIN Web site can be found in Appendix K.

The effect of coverage threshold alteration is similar to the effect of precision threshold alteration. When the coverage threshold value is 0, then the value of coverage-with-threshold

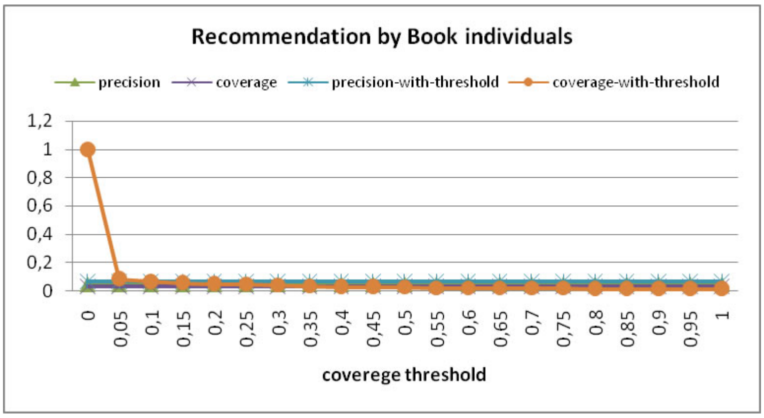

Figure 4.23: Book class, Minimum Support Threshold: 0.575% , SECKIN Web Site

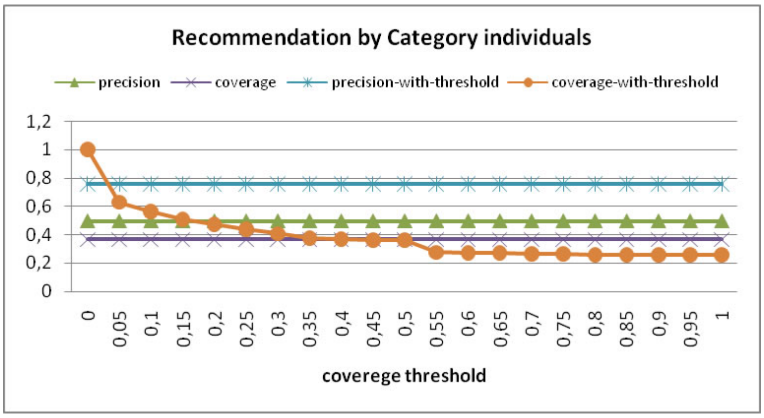

Figure 4.24: Category class, Minimum Support Threshold: 4.5% , SECKIN Web Site

is 1. When the coverage threshold value starts to increase then the value of coverage-with-threshold starts to drop. If the coverage value is far from 1, then a sharp decrease occurs (as in the case of Author and Book); on the other hand, a steady decrease occurs if the coverage value is relatively closer to 1 (as in the case of PublishYear, Vendor and Category).

When the coverage threshold reaches 1, then the value of coverage-with-threshold reaches its minimum. At that point, new coverage of the very few sessions whose recommendations are all accurate will be 1.

At a breaking point, the value of coverage-with-threshold and actual coverage values are equal (i.e. 0.35 for Category, 0.45 for PublishYear and 0.3 for Vendor). After that point, the coverage-with-threshold shows a steadier decrease until the coverage threshold value reaches 1.

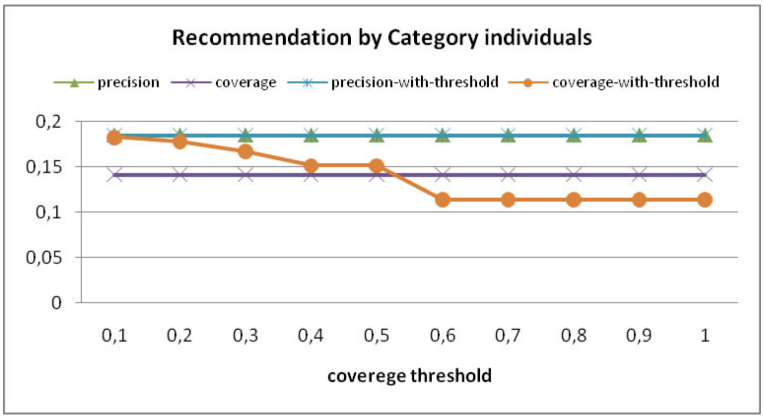

Figure 4.25: Category class, Minimum Support Threshold: 0.6% , CENG Web Site

Figures 4.25 display the evaluation results in terms of the Category class.

This evaluation shows similarity with that of the SECKIN web site. The coverage-with-threshold value shows a steady decrease.

In Appendix L effect of precision threshold in terms of classes of CENG web site is presented. Similar to precision-with-threshold, when the coverage threshold value is 0, then the value of coverage-with-threshold is 1. Since the coverage of the Author and Thread class is very small, a sharp drop occurs in the coverage-with-threshold values of these classes. On the other hand, in the evaluation of the recommendation with Group class, a steady decrease occurs since the average precision value in this evaluation is higher.

4.5 Evaluation of the effect of number of association rules

When making a recommendation, an association rule that is similar to the user navigational pattern is sought. This search requires the iteration of all the association rules found earlier. In some circumstances, there can be too many association rules (especially in cases of low minimum support and minimum confidence) and their iteration can require quite a considerable time. In this evaluation, the effect of eliminating some of the association rules on the accuracy and timing of the recommendation is studied.

In the first test group, the evaluation result is shown in terms of the eliminated association rule percentage. In the other test group, the time required to make the recommendations is shown.

4.5.1 Effect of number of association rules on recommendation quality

The following charts show the relation between precision, coverage, new precision, new coverage and association rule percentage used for recommendation. The limiting of the association rule is done by taking only the first n percentage association rules. The following parameters are used for each evaluation.

Minimum Confidence :10%

Window Count:1

Threshold Precision:0.3

Threshold Coverege:0.3

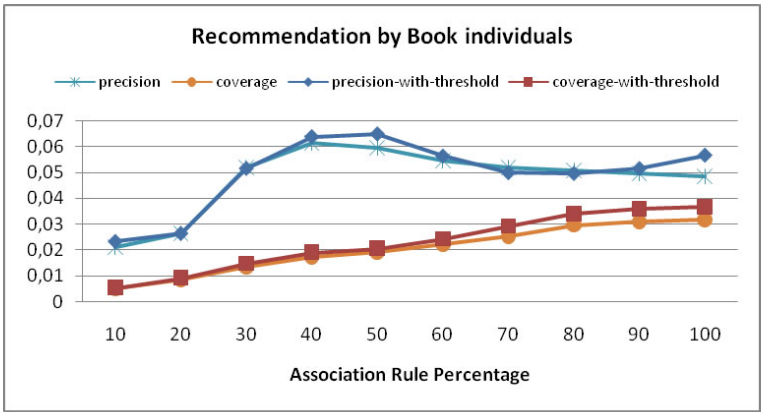

Figure 4.26: Book class, Minimum Support Threshold: 0.575% , SECKIN Web Site

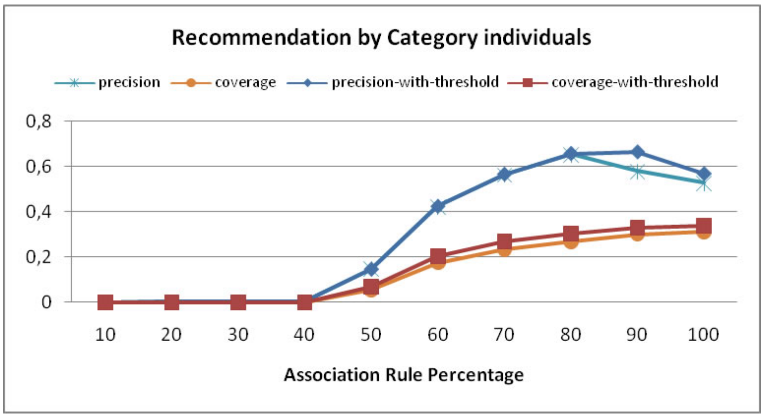

Figure 4.27: Category class, Minimum Support Threshold: 4.5% , SECKIN Web Site

Figures 4.26, and 4.27 display the evaluation results in terms of the Book class, Category class respectively. Evalution in terms of other classes of SECKIN Web site can be found in Appendix M.

The precision value increases when the association rule percentage increases. This is the expected result since the number of recommendations will increase by increasing association rules; the accurate recommendation among them also increased as well. However, after a certain percentage, the precision starts to drop. The reason for this is increasing the number of association rules also increases the number of accurate as well as inaccuare recommendation. However, after a breaking point, the amount of inaccurate recommendations increases faster than that of accurate recommendations. Hence, the precision value starts to decrease.

The coverage value increases steadily. The reason for this is that the number of covered items in the evaluation part of each session increases when the number of recommendations increases.

In Appendix N effect of number of association rules on recommendation quality in terms of classes of CENG web site is presented. Since the number of association rules is very low in the evaluation of the recommendations of the Author and Thread class, the precision value only exists when the rule percentage is higher than 90%.

In the graph of the evaluation of the Group class, the precision and coverage values show a steady increase. Between 60% and 90%, the precision value does not change; between 90% and 100% the precision value increases.

4.5.2 Effect of number of association rules on time performance

The following charts show the relation between time and the association rule percentage used for recommendation. The time includes the total amount of time elapsed for producing recommendations for each session at the evaluation part of the dataset. It does not include the association rule finding times or the evaluation times. The following parameters are used for each evaluation.

Minimum Confidence :10%

Window Count:1

Threshold Precision:0.3

Threshold Coverege:0.3

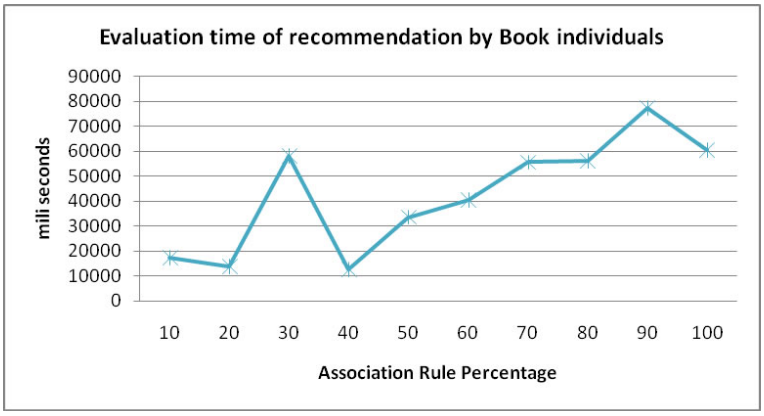

Figure 4.28: Book class, SECKIN Web Site

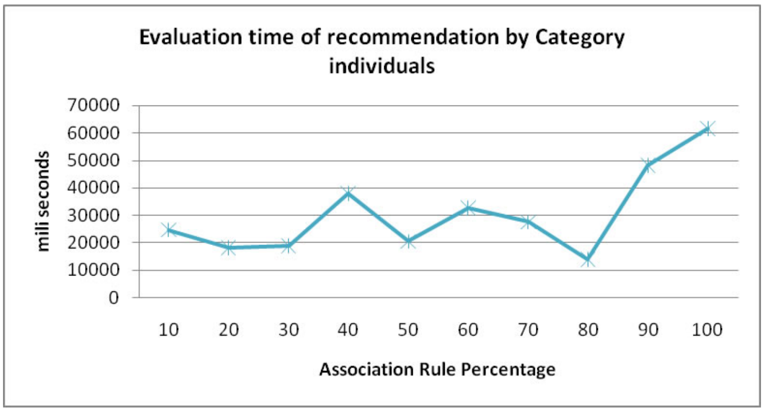

Figure 4.29: Category class, SECKIN Web Site

Figures 4.28, and 4.29 display the elapsed time for the recommendation with the different percentages of the used association rules. The graphs of other classes can be found in Appendix O.

Establishing accurate elapsed time for the recommendation is very hard since the evaluation process in done in a multi-threaded operation system, which means at the same time there are many tasks to do for the computer. In particular, the evaluation program and database server are in the same computer, and these two processes consume a great deal of memory and CPU time when the evaluation is running, so the evaluation time would vary greatly from run to run.

To obtain an average elapsed time, the same evaluation is made 6 times. For each run, the minimum and maximum elapsed times are pruned and the average of the other four elapsed times is calculated to be the elapsed time.

The elapsed time generally increases when the percentage of the association rules increases. This is expected since the more association rules to look up for the recommendation there are, the more time is required. However, the elapsed time does not show a linear but a traverse increase. Because the workload of the computer may differ from test to test, the elapsed time of some tests is higher than the real elapsed time, and some are lower; therefore, the line show a traverse increase.

There is a trade-off between time and success of the recommendation. The more association rules are used, the more time elapses in contrast when the number of association rules that are used for making recommendations is low then the precision and coverage value of the recommendation are also low.

It can be extracted from the above evaluations that, if the time is limited for recommendation, then up to 30% of the association rules can be eliminated to reduce the time of the recommendation, and the recommendations will still be within the acceptable range.

4.6 Brief Conclusion of Evaluations

Including semantic information generates recommendations with more precision and coverage values than recommendation without semantic information. However, it should be emphasized that, not every ontology class increases the precision and coverage values. For example, the evaluations in terms of the Author and Thread classes in the CENG web site do not have as high precision and coverage values as the evaluations in terms of the Group class. The number of individuals is an important criterion when choosing the classes to use in recommendation. If the number of individuals is low, then more accurate recommendations are produced (high precision value) and the recommendation covers more items that the user is likely to visit (high coverage value).

In addition, including semantic information allows the generation of recommendations at higher minimum support and minimum confidence threshold values when compared to the recommendations made in terms of web page addresses; and also the number of freqent sequences is higher in web usage mining with semantic information.

The window count parameter does not generally improve the precision and coverage values. This shows that the recommendations are generally made, using only the last item of the recommendation part of the session, and the recommendations that use more than one of the last items are very rare.

Moreover, the evaluations shows the trade-off between time and success of the recommendation. The more association rules used, the more time elapses but the more recommendation accuracy is.

CHAPTER 5

CONCLUSION

5.1 Conclusion

The extracting of usage patterns or the finding of common usage profiles of the visitors is highly important for web site developers. The results can be used in many areas, such as recommendations of the pages the user would probably visit at the next step, recommending to the visitor a product to buy, or content improvement, or page displacement. Besides this, the result of web usage mining can be helpful to the web site owner in decision making, such as which content group of a web site should be enriched, if it is a shopping web site, which products should be promoted, or page replacement. Currently, web usage mining algorithms are used to extract the common navigational patterns of the visitors.

One of the biggest disadvantages of web usage mining is that the results are in the form of web pages; hence, there is no semantic meaning of the common navigation profile. Site owners generally want to learn which category or group is more popular, which page is visited more than others. To get such results, the results of web usage mining have to be analyzed by someone, and this process can be both erroneous and time-consuming.

Another disadvantage of classical web usage mining is called the new-item problem, which is the failure to recommend newly added pages or products to the visitors since these products or pages are not in the current common navigation profiles.

In this thesis, the disadvantages of classical web usage mining are overcome by including semantic information in web usage mining. By introducing semantic information, web usage mining algorithms are performed in terms of ontology individuals instead of web page addresses. A different approach is proposed that seperately finds the frequent navigation paths in terms of all distinct properties of a particular ontology class. The frequent navigation paths are used to generate recommendation for the user. Recommendations either are generated considering the frequent navigation paths in terms of a single class or considering combinition of frequent navigation paths in terms of a several classes.

The system consists of 3 phases. The first phase is the preprocessing step. In this step, the unnecessary entries from web server logs are pruned and the navigation histories of each visitor to the web site is extracted. The navigation history is constructed as a series of ontology individuals instead of a series of web page addresses. The next phase is the rule extraction step. In this phase, for each class of the ontology, association rules are extracted in terms of the class's individuals.

The SPADE [7] algorithm is used to extract the common navigation paths of the web users. Some optimizations are made to the SPADE algorithm, such as pruning uninteresting frequent sequences after finding the frequent sequences. Also, the SPADE algorithm is extended to handle the taxonomy of the items like the GSP algorithm.

The third and last phase is the recommendation phase. In this phase, the association rules are joined with the user's online navigation path, and new pages are recommended to the user.

The success of the system is measured by the evaluation of the recommendations. Evaluation is done by performing a 10-fold cross-validation on 3 datasets. The precision and coverage values of the recommendations are measured for each session in the evaluation data set. Moreover, two new measurements called new precision and new coverage are introduced. By changing certain parameters and keeping the other parameters constant, many tests are performed on the datasets especially the SECKIN web site.

The result of the evaluation show that including semantic information produces stronger and more confident association rules. For example, the association rules can be extracted at relatively high minimum support in terms of individuals of the Category class; on the other hand, to extract association rules in terms of web page addresses or in terms of individuals of the Book class, the algorithm should run on a low support percentage which requires considerable CPU time.

The new-item problem of web usage mining entails not recommending the newly added items to the users, since a substantial number of visits have to be made to the newly added item, for it be recommended. The proposed system shows that when semantic information is included, the new-item problem of web usage mining is overcome since the recommendations are based on the semantic meaning of the frequent navigation paths. One of the reasons for the increase in the precision and coverage value of the recommendations in terms of semantic information is the capability to recommend the newly-added items to the users.

The association rule finding can be done in terms of a single class individual, and also in terms of the combination of all the individuals of the ontology classes. The combined association rules have greater precision value and coverage value than the classical web usage mining, as shown in the evaluation section of the thesis.

5.2 Future Work

There are many projects or tasks possible to improve the proposed system. One project is including the full power of the ontology. In the thesis, the web sites that are used to extract association rules do not have semantic information. Therefore the ontologies of the web sites are extracted by manual operations. Although, some taxomomies are used between the individuals of some classes in the ontologies, the ontologies are kept simple due to the limited time of the project. Therefore, the power of ontology such as complex hierarchies among classes or complex properties is not defined in the ontologies.

Another project is introducing time weight to the generated frequent paths. Frequent navigation paths that are generated from recent navigations has more effect to the recommendations than freqeunt navigation paths of the old visit log. Therefore a time weight parameter can be introduced to emphasize more to the recent frequent navigation paths.

The future projects concerning web usage mining of the Semantic Web would include ontology reasoning and the inference power of the ontology. Hence, stronger and more confident rules can be extracted from a web site and the semantic meaning of the common usage profile would be clearer.

Another interesting project would be to employ web usage mining on semantic usage logs. Today, all web servers register users visit information in a non-semantic way. Neither information about visited class individuals nor any ontology related knowledge is presented in the web server logs. Therefore, Web usage mining is carried out by converting non-semantic web usage logs into semantic information; this makes the process more time consuming and some interesting rules may be eliminated in the process of semantic information conversion.

In the thesis, the association rule mining algorithms are used to extract common navigational patterns which are those constructed in sequential forms. Also, there are some other data mining methods that can be used to find common usage patterns. It would be an interesting subject in the future to use clustering or classification algorithms to extract semantic common usage profiles.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In Proc. of the ACM SIGMOD Conference on Management of Data, pages 207-216, Washington, D.C., May 1993.

[2] R. Agrawal, R. Srikant: "Fast Algorithms for Mining Association Rules", Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, Sept. 1994. Expanded version available as IBM Research Report RJ9839, June 1994.

[3] R. Srikant, R. Agrawal: "Mining Generalized Association Rules", Proc. of the 21st Int'l Conference on Very Large Databases, Zurich, Switzerland, September 1995. Expanded version available as IBM Research Report RJ 9963, June 1995.

[4] Bettina Berendt, Andreas Hotho and Gerd Stumme: Towards Semantic Web Mining. In International Semantic Web Conference (ISWC) Springer. 2002, 264-278.

[5] G. Stumme, B. Berendt, A. Hotho: Usage Mining for and on the Semantic Web. Next Generation Data Mining. Proc. NSF Workshop, Baltimore, Nov. 2002, 77-86.

[6] Honghua Dai, Bamshad Mobasher: Integrating Semantic Knowledge with Web Usage Mining for Personalization. In Web Mining: Applications and Techniques, Anthony Scime (ed.), IRM Press, Idea Group Publishing. 2005.

[7] M. J. Zaki. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Learning Journal, 42, 3160

[8] R. Srikant, R. Agrawal: "Mining Sequential Patterns: Generalizations and Performance Improvements", to appear in Proc. of the Fifth Int'l Conference on Extending Database Technology (EDBT), Avignon, France, March 1996. Expanded version available as IBM Research Report RJ 9994, December 1995.

[9] R. Agrawal, R. Srikant: "Mining Sequential Patterns", Proc. of the Int'l Conference on Data Engineering (ICDE), Taipei, Taiwan, March 1995. Expanded version available as IBM Research Report RJ9910, October 1994.

[10] R. Agrawal, T. Imielinski, A. Swami: "Mining Associations between Sets of Items in Massive Databases", Proc. of the ACM-SIGMOD 1993 Int'l Conference on Management of Data, Washington D.C., May 1993.

[11] A. Hotho, A. Maedche, S. Staab, and R. Studer. SEAL-II the soft spot between richly structured and unstructured knowledge. Journal of Universal Computer Science (J.UCS), 7(7):566590, 2001.

[12] D. Oberle. Semantic Community Web Portals -Personalization, Studienarbeit. Universit¨at Karlsruhe, 2000.

[13] B. Mobasher, H. Dai, T. Luo, Y. Sun, and J. Zhu. Integrating web usage and content mining for more effective personalization. In Proceedings of the International Conference on E-Commerce and Web Technologies (ECWeb2000), pages 165176, Greenwich, UK, 2000.

[14] H. Dai and B. Mobasher. Using ontologies to discover domain-level web usage profiles. In Proceedings of the Second Semantic Web Mining Workshop at PKDD 2001, km.aifb.uni-karlsruhe.de/semwebmine2002/papers/full/bamshad.pdf, Aug 2002.

[15] J. L. Borges and M. Levene. Data mining of user navigation patterns. In [49], pages 92111. 2000.

[16] M. Baumgarten, A.G. B¨uchner, S.S. Anand, M.D. Mulvenna, and J.G. Hughes. User-driven navigation pattern discovery from internet data. In [49], pages 74 91. 2000.

[17] M. Spiliopoulou. The laborious way from data mining to web mining. International Journal of Computer Systems, Science, Engineering, 14:113126, 1999.

[18] M. Spiliopoulou, C. Pohle, and M. Teltzrow. Modelling and mining web site usage strategies. In Proceedings of the Multi-Konferenz Wirtschaftsinformatik, Sep 2002.

[19] M. Spiliopoulou and C. Pohle. Data mining for measuring and improving the success of web sites. Data Mining and Knowledge Discovery, 5:8514, 2001.

[20] Charu C. Aggarwal. Collaborative Crawling: Mining User Experiences for Topical Resource Discovery In IBM Research Report , ACM, pages 423428, 2002.

[21] T. Joachims. Optimizing search engines using clickthrough data. In [21], pages 133142, 2002.

[22] C. Kemp and K. Ramamohanarao. Long-term learning for web search engines. In Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2002), pages 263274, Berlin, 2002. Springer.

[23] P. Melville, R.J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering. In Proceedings of the ACM SIGIR Workshop on Recommender Systems, Sep 2001.

[24] J.J. Sandvig, B. Mobasher, and R. Burke. Robustness of Collaborative Recommendation Based On Association Rule Mining. Proceedings of the 2007 ACM Conference on Recommender Systems, Minneapolis, October, 2007.

[25] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation. Data Mining and Knowledge Discovery 8:53-87, 2004.

[26] Mohammed J. Zaki. Scalable algorithms for association mining. IEEE Transactions on Knowledge and Data Engineering, 12(3):372-390, May/June 2000.

[27] Honghua (Kathy) Dai, Bamshad Mobasher. A Road map to More Effective Web Personalization: Integrating Domain Knowledge with Web Usage Mining . In Proceedings of the International Conference on Internet Computing 2003 (IC03), Las Vegas, Nevada, June 2003.

[28] Miki Nakagawa, Bamshad Mobasher. Impact of Site Characteristics on Recommendation Models Based On Association Rules and Sequential Patterns. In Proceedings of the IJCAI'03 Workshop on Intelligent Techniques for Web Personalization, Acapulco, Mexico, August 2003.

[29] Ahu Sieg, Bamshad Mobasher, Robin Burke, Ganesh Prabu, Steve Lytinen. Representing User Information Context with Ontologies. In Proceedings of HCI International 2005 Conference, Las Vegas, Nevada, July 2005.

[30] B. Ganter and G. Stumme. Creation and merging of ontology top-levels. In Proc. ECAI02. submitted, 2002.

[31] Gediminas Adomavicius and Er Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, pages 734–749, 2005 Volume 17.

[32] G. Linden, B. Smith, and J. York, Amazon.com Recommendations: Item-to-Item Collaborative Filtering, IEEE Internet Computing, Jan./Feb. 2003

[33] G. Salton, Automatic Text Processing. Addison-Wesley, 1989

[34] World Wide Web Consortium (W3C), from http://www.w3.org/

[35] Resource Description Framework (RDF) Model and Syntax Specification, from http://www.w3.org/TR/PR-rdf-syntax/

[36] W3C working group, from http://www.w3.org/2007/OWL

[37] TopQuadrant Technology Briefing, Semantic Technology, Version 1.2 , March 2004

[38] Protégé, from http://protege.stanford.edu/

[39] UCI Knowledge Discovery in Databases Archive, from http://kdd.ics.uci.edu/

APPENDIX A

DATABASE DIAGRAMS

Figure A.1: Database diagram of the table related with frequent sequence finding

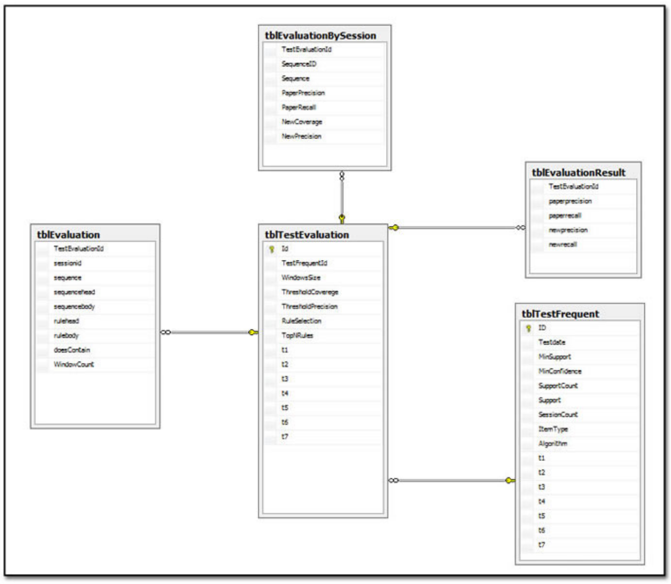

Figure A.2: Database diagram of the table related with evaluation

APPENDIX B

TIME COMPARISION OF GSP AND SPADE ALGORITHMS

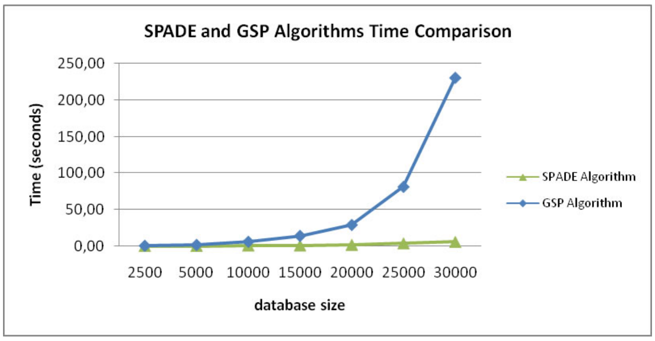

Figure B.1: Time Comparision of GSP and SPADE Algorithms

The Figure B.1 shows the time relation between GSP and SPADE algorithms. Database size refers to the total number of items in all sequences of the database.

APPENDIX C

ONTOLOGY DEFINITIONS

Since the ontology files needs many pages to display, the files have been placed on the Internet. From the following web page address, the ontologies of each web site can be downloaded.

http://sites.google.com/site/e1522648/Home

APPENDIX D

ASSOCIATION RULES SAMPLES

This section contains some sample association rules.

Association in term of individuals of Book classes in SECKIN Web Site

Rule	Confidence
Terazi Aylık Hukuk Dergisi Yıl: 3 Sayı: 17 Ocak 2008 ⇒ Takip Hukukunda Zamanaşımı	0.176529589
Terazi Aylık Hukuk Dergisi Yıl: 3 Sayı: 17 Ocak 2008 ⇒ Terazi Aylık Hukuk Dergisi Yıl: 2 Sayı: 14 Ekim 2007	0.151454363
Terazi Aylık Hukuk Dergisi Yıl: 3 Sayı: 17 Ocak 2008 ⇒ Karayolları Trafik Kanununa Göre...Trafik Kabahatleri	0.16449348
İcra ve İflas Hukuku ⇒ İcra ve İflas Hukuku	0.312244898
Terazi Aylık Hukuk Dergisi Yıl: 3 Sayı: 17 Ocak 2008 ⇒ Medeni Usul Hukukunda Davanın Açılmamış Sayılması	0.153460381
Terazi Aylık Hukuk Dergisi Yıl: 3 Sayı: 17 Ocak 2008 ⇒ Terazi Aylık Hukuk Dergisi Yıl: 2 Sayı: 16 Aralık 2007	0.153460381
Terazi Aylık Hukuk Dergisi Yıl: 3 Sayı: 17 Ocak 2008 ⇒ Delillerin Gösterilmesi ve Toplanmasında Kesin Süre	0.168505517
Terazi Aylık Hukuk Dergisi Yıl: 3 Sayı: 17 Ocak 2008 ⇒ Terazi Aylık Hukuk Dergisi Yıl: 2 Sayı: 13 Eylül 2007	0.155466399
Terazi Aylık Hukuk Dergisi Yıl: 3 Sayı: 17 Ocak 2008 ⇒ Adli Bilimler Dergisi - Cilt: 6 Sayı: 3 Eylül 2007	0.151454363
Terazi Aylık Hukuk Dergisi Yıl: 3 Sayı: 17 Ocak 2008 ⇒ Terazi Aylık Hukuk Dergisi Yıl: 3 Sayı: 17 Ocak 2008	0.231695085

The following values are used in the above experiment; Minimum Support Threshold : 1.1 %, and Minimum Confidence Threshold : 10 %

Association in term of individuals of Category classes, SECKIN Web Site

Rule	Confidence
Bilgisayar ⇒ Bilgisayar	0.401987353206865
İşletme ⇒ İşletme	0.41125703564728
Sınavlara Hazırlık ⇒ Sınavlara Hazırlık	0.364384463462804
Hukuk ⇒ Hukuk→Hukuk	0.247158578387155
Hukuk ⇒ Hukuk	0.408082265920981
Hukuk, Süreli Yayınlar ⇒ Hukuk	0.760741364785173
Hukuk→Hukuk ⇒ Hukuk	0.605658709106985
Hukuk→Hukuk ⇒ Hukuk→Hukuk→Hukuk	0.359858532272325
Hukuk→Hukuk ⇒ Hukuk→Hukuk	0.451370468611848
Hukuk, Süreli Yayınlar, Terazi Hukuk Dergisi ⇒ Hukuk	0.789330922242315
Hukuk→Hukuk→Hukuk ⇒ Hukuk	0.745255474452555
Hukuk→Hukuk→Hukuk ⇒ Hukuk→Hukuk	0.594160583941606
Hukuk→Hukuk→Hukuk→Hukuk ⇒ Hukuk	0.797257590597453

The following values are used in the above experiment; Minimum Support Threshold : 5.5 %, and Minimum Confidence Threshold : 20 %

Association in term of individuals of Group classes, CENG Web Site

Rule	Confidence
metu.ceng.test ⇒ metu.ceng.test	0.886237831647261
metu.ceng.test ⇒ metu.ceng.test → metu.ceng.test	0.79795762550105
metu.ceng.test ⇒ metu.ceng.test → metu.ceng.test → metu.ceng.test	0.722275243367055
metu.ceng.test → metu.ceng.test ⇒ metu.ceng.test	0.900387680379065
metu.ceng.test → metu.ceng.test ⇒ metu.ceng.test → metu.ceng.test	0.814990307990523
metu.ceng.test → metu.ceng.test → metu.ceng.test ⇒ metu.ceng.test	0.905154885779213

The following values are used in the above experiment; Minimum Support Threshold : 24 %, and Minimum Confidence Threshold : 10 %

APPENDIX E

EFFECT OF MINIMUM SUPPORT THRESHOLD USING SINGLE CLASS RULE, SECKIN WEB SITE

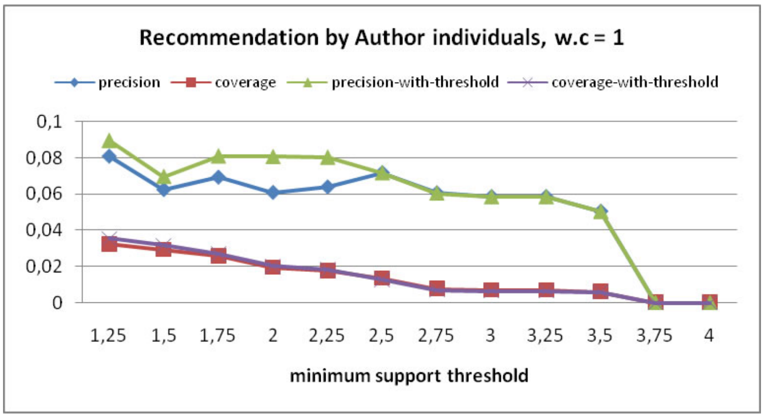

Figure E.1: Author class, window count = 1, SECKIN Web Site

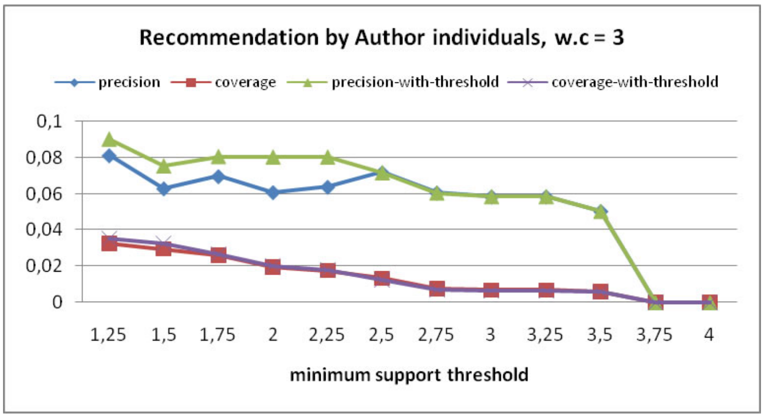

Figure E.2: Author class, window count = 3, SECKIN Web Site

Figures E.1 and E.2 display the evaluation results in terms of the Author class and their window count parameters are 1 and 3 respectively. The number of distinct Author individuals is 4,545. Since the number of Book individuals and the number of Author individuals are relatively high, the precision and coverage values of the evaluations show similar behaviors. Association rules are extracted with only a small minimum support count and the precision and coverage values are relatively small.

Similar to the evaluation of the Book class, the time of the evaluation gets extremely long when the minimum support count gets smaller. It is not feasible to test with a minimum support threshold less than 1.25%.

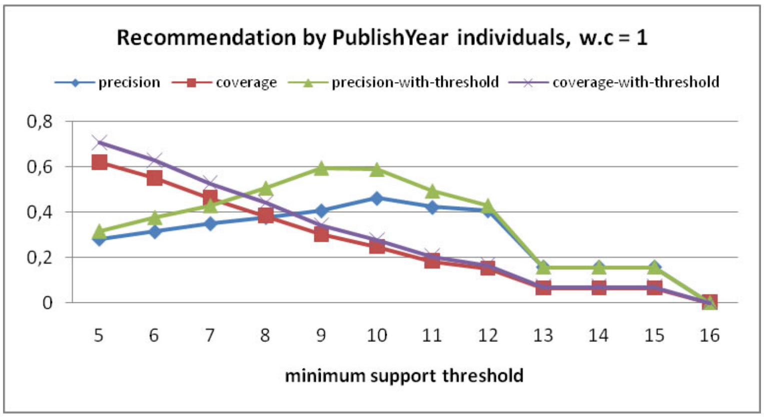

Figure E.3: PublishYear class, window count = 1, SECKIN Web Site

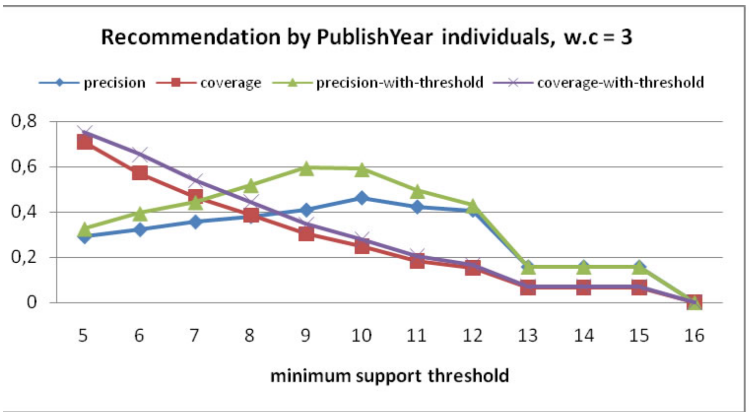

Figure E.4: PublishYear class, window count = 3, SECKIN Web Site

Figures E.3 and E.4 display the evaluation results in terms of the PublishYear class and their window count parameters are 1and 3 respectively. The number of distinct PublishYear individuals is 34.

Since the number of individuals is small, as in the Category class, the evaluation results show similar behaviors with the evaluation of the Category class. At first, the precision value increases when the minimum support threshold increases, since the number of the association rules decreases when the minimum support threshold increases. The breaking point of the precision value (10%) is higher than that of Categories' because, at a given minimum support threshold, more association rules are produced in the PublishYear evaluation than the Category evaluation. After the breaking point, the precision values start to decrease since the number of sessions whose precision value is 0 increases.

The coverage value decreases constantly when minimum support increases since the number of covered items in the session's evaluation part decreases when the minimum support count increases, so the coverage value always decreases.

As in the Category class, at small minimum support threshold values, increasing the window count also increases precision and coverage values a little, but after a threshold value (minimum support count = 7%) , the window count does not affect precision and coverage values.

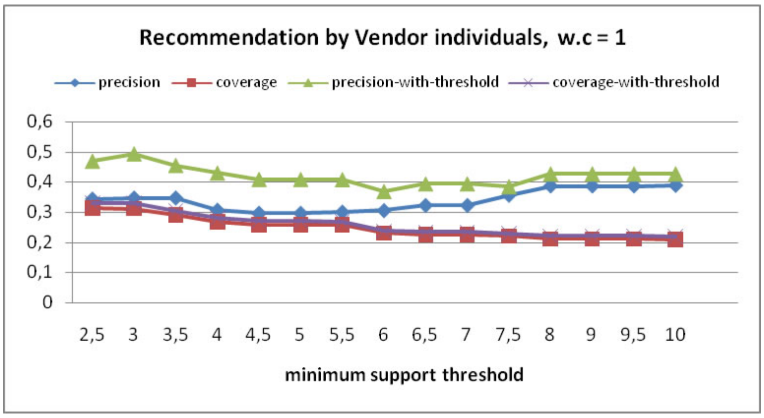

Figure E.5: Vendor class, window count = 1, SECKIN Web Site

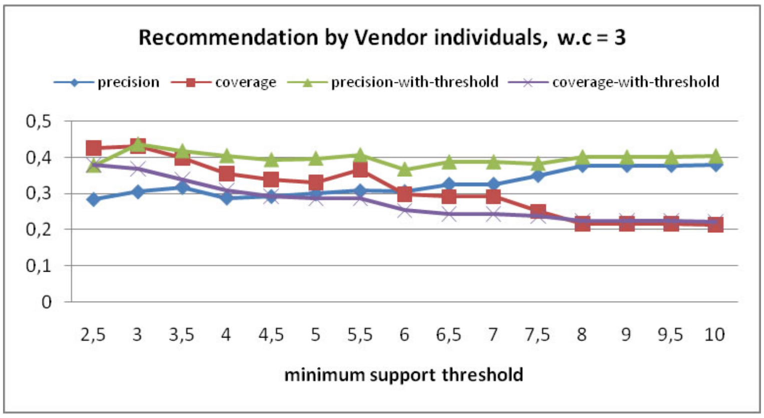

Figure E.6: Vendor class, window count = 3, SECKIN Web Site

Figures E.5, and E.6 display the evaluation results in terms of the Vendor class and their window count parameters are 1 and 3 respectively. The number of distinct Vendor individuals is 249.

Since the number of distinct individuals is relatively small, as in the Category and PublishYear class the evaluation results show similar behaviors with the evaluation in terms of the Category and PublishYear classes. The precision values increase when the minimum support count increases, because when the minimum support increases, the number of association rules decreases. The breaking point has not been reached in the charts; although the precision value increases at quite a small amount after the 8% minimum support threshold, it does increase. Therefore, most probably, the breaking point for the precision value is a little greater than 10%.

Coverage value decreases constantly when minimum support increases, because the number of covered items in the session's evaluation part decreases when the minimum support increases, so coverage value always decreases.

APPENDIX F

EFFECT OF MINIMUM SUPPORT THRESHOLD USING SINGLE CLASS RULE, CENG WEB SITE

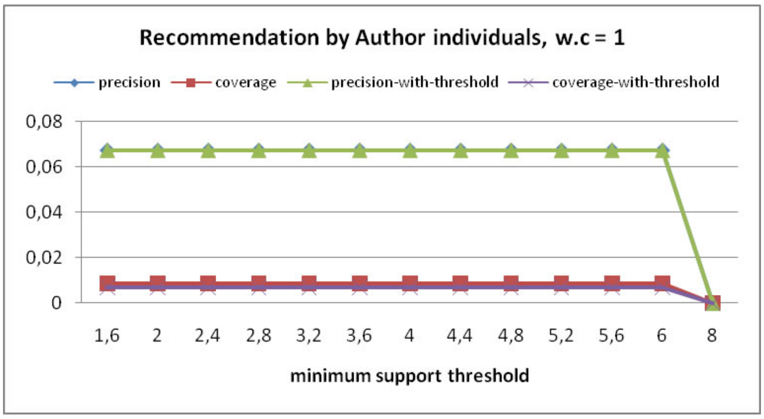

Figure F.1: Author class, window count = 1, CENG Web Site

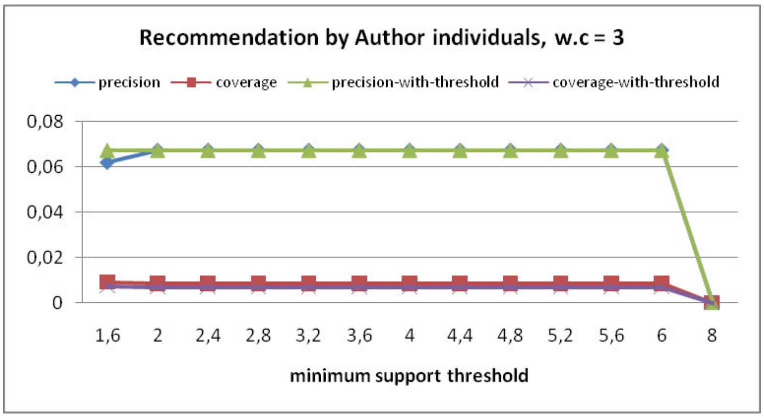

Figure F.2: Author class, window count = 3, CENG Web Site

Figures F.1, and F.2 show the evaluation results in terms of the Author class, and their window count parameters are 1 and 3 respectively. The number of distinct Author individuals is 500. Association rules can be found until minimum support 8, so after 8 the precision and coverage values drop to 0, since no association rules are generated to make recommendations. This test shows similar characteristic to the MSNBC data set; that is, changing the minimum support does not change the number of association rules considerably. Therefore, the precision and coverage values do not change below minimum support 6.

Similar to the Thread class, changing the window count parameter does not affect the precision and coverage values.

APPENDIX G

EFFECT OF MINIMUM CONFIDENCE THRESHOLD USING SINGLE CLASS RULE, SECKIN WEB SITE

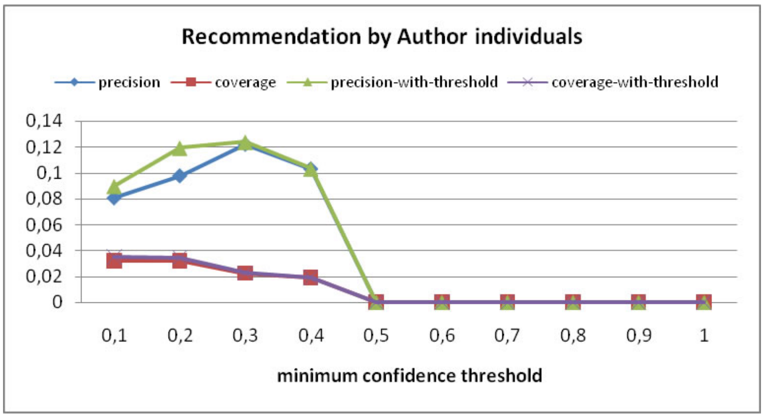

Figure G.1: Author class, Minimum Support Threshold: 1.25% , SECKIN Web Site

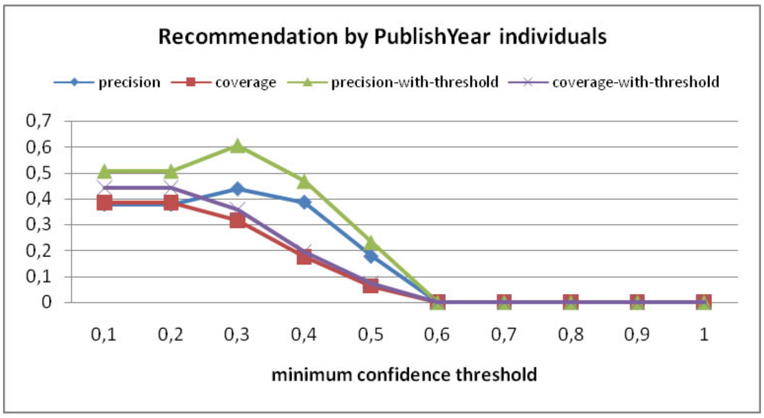

Figure G.2: PublishYear class, Minimum Support Threshold: 8% , SECKIN Web Site

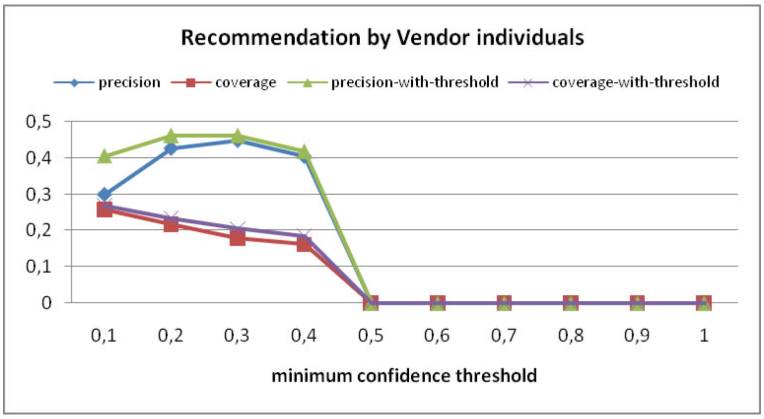

Figure G.3: Vendor class, Minimum Support Threshold: 8% , SECKIN Web Site

APPENDIX H

EFFECT OF MINIMUM CONFIDENCE THRESHOLD USING SINGLE CLASS RULE, CENG WEB SITE

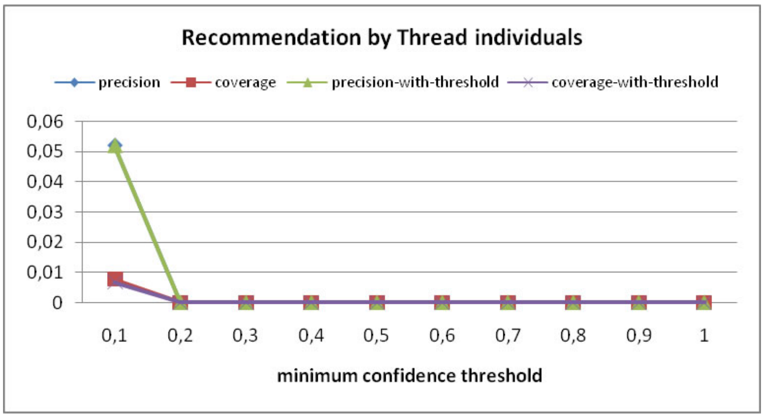

Figure H.1: Thread class, Minimum Support Threshold: 2% , CENG Web Site

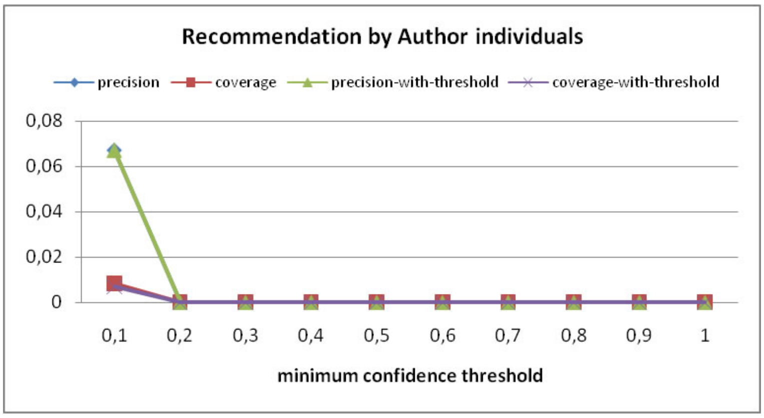

Figure H.2: Author class, Minimum Support Threshold: 2% , CENG Web Site

APPENDIX I

EFFECT OF PRECISION THRESHOLD, SECKIN WEB SITE

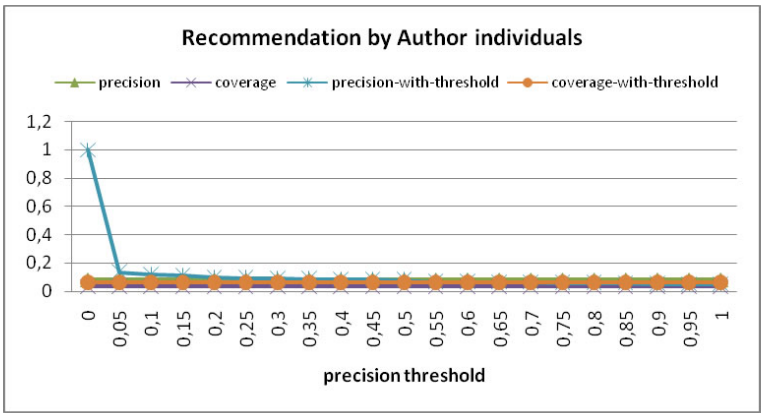

Figure I.1: Author class, Minimum Support Threshold: 1.25% , SECKIN Web Site

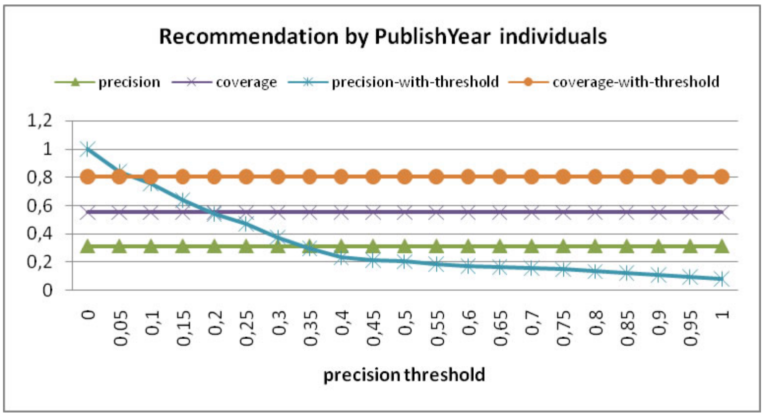

Figure I.2: PublishYear class, Minimum Support Threshold: 6% , SECKIN Web Site

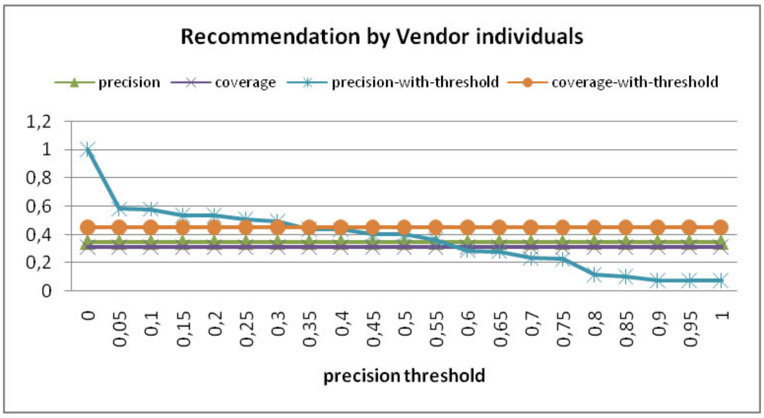

Figure I.3: Vendor class, Minimum Support Threshold: 3% , SECKIN Web Site

APPENDIX J

EFFECT OF PRECISION THRESHOLD, CENG WEB SITE

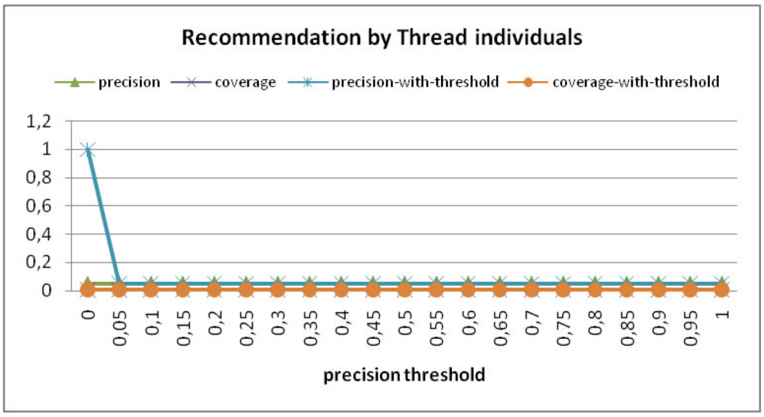

Figure J.1: Thread class, Minimum Support Threshold: 2% , CENG Web Site

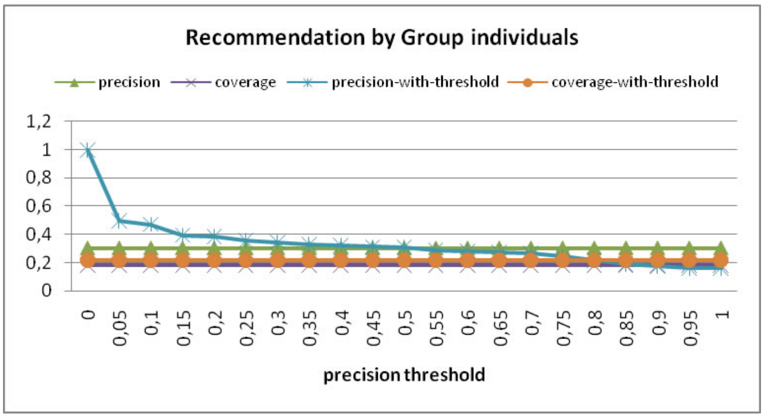

Figure J.2: Group class, Minimum Support Threshold: 8% , CENG Web Site

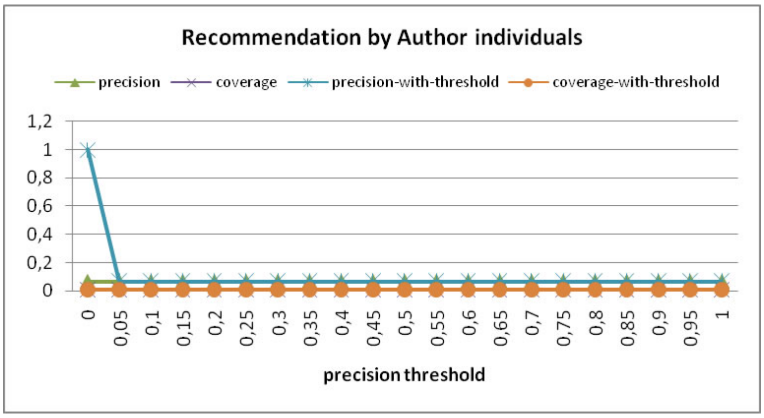

Figure J.3: Author class, Minimum Support Threshold: 2% , CENG Web Site

APPENDIX K

EFFECT OF COVERAGE THRESHOLD, SECKIN WEB SITE

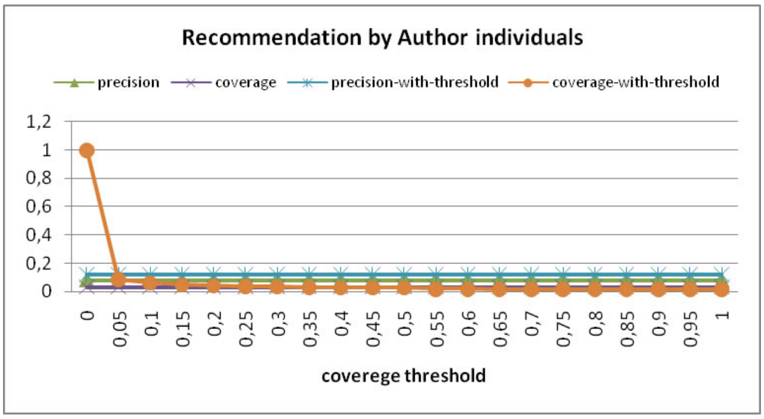

Figure K.1: Author class, Minimum Support Threshold: 1.25% , SECKIN Web Site

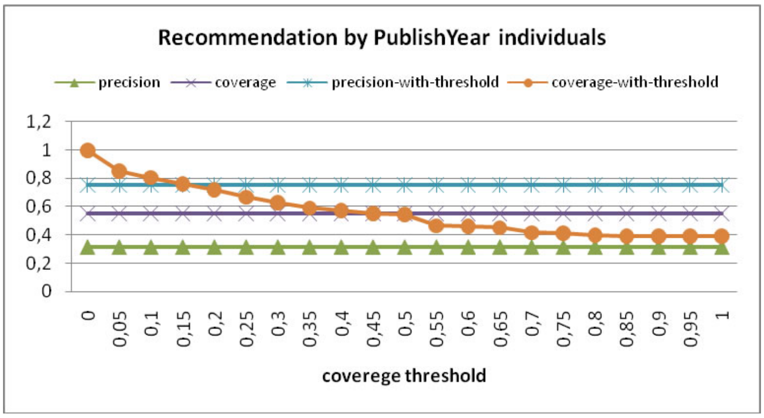

Figure K.2: PublishYear class, Minimum Support Threshold: 6% , SECKIN Web Site

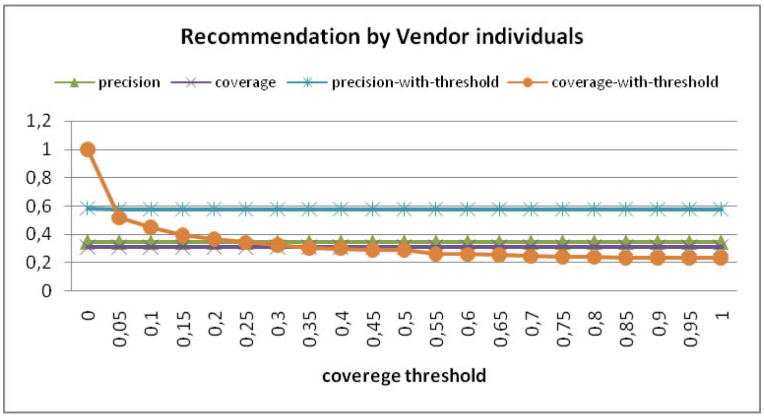

Figure K.3: Vendor class, Minimum Support Threshold: 3% , SECKIN Web Site

APPENDIX L

EFFECT OF COVERAGE THRESHOLD, CENG WEB SITE

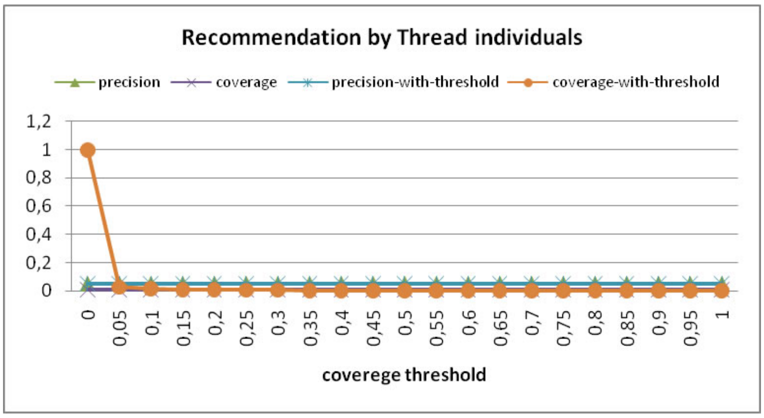

Figure L.1: Thread class, Minimum Support Threshold: 2% , CENG Web Site

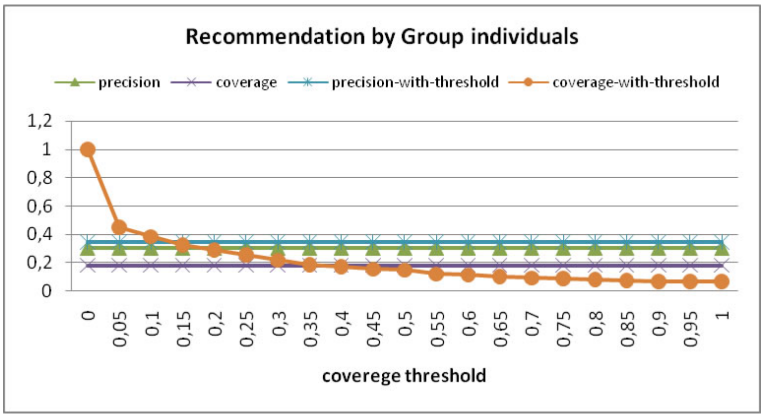

Figure L.2: Group class, Minimum Support Threshold: 8% , CENG Web Site

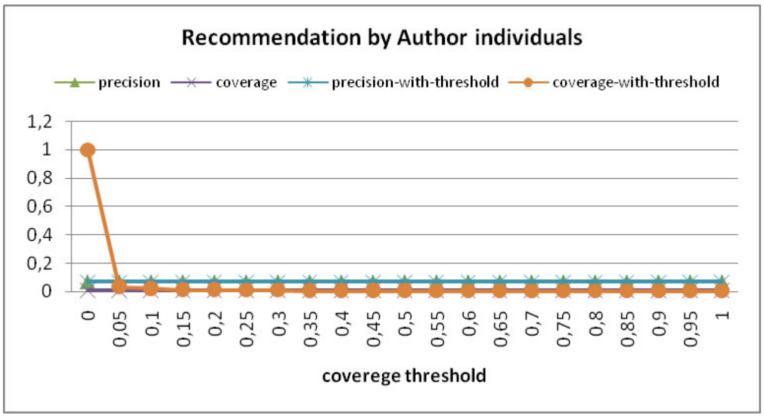

Figure L.3: Author class, Minimum Support Threshold: 2% , CENG Web Site

APPENDIX M

EFFECT OF NUMBER OF ASSOCIATION RULES ON RECOMMENDATION QUALITY, SECKIN WEB SITE

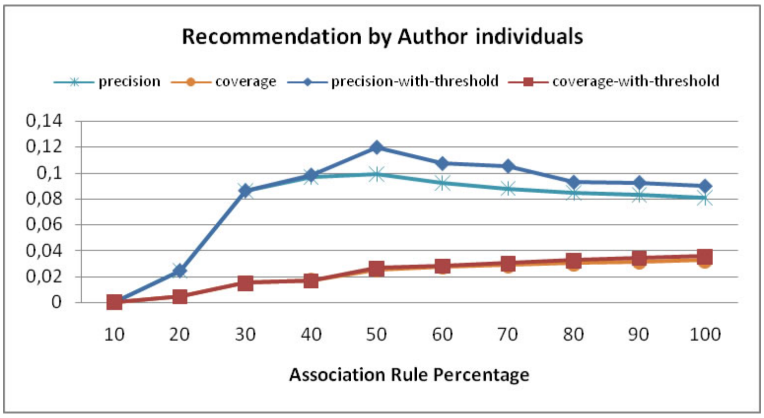

Figure M.1: Author class, Minimum Support Threshold: 1.25% , SECKIN Web Site

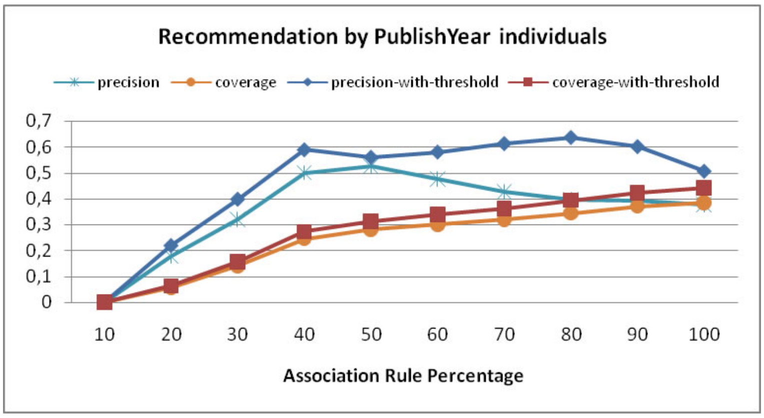

Figure M.2: PublishYear class, Minimum Support Threshold: 6% , SECKIN Web Site

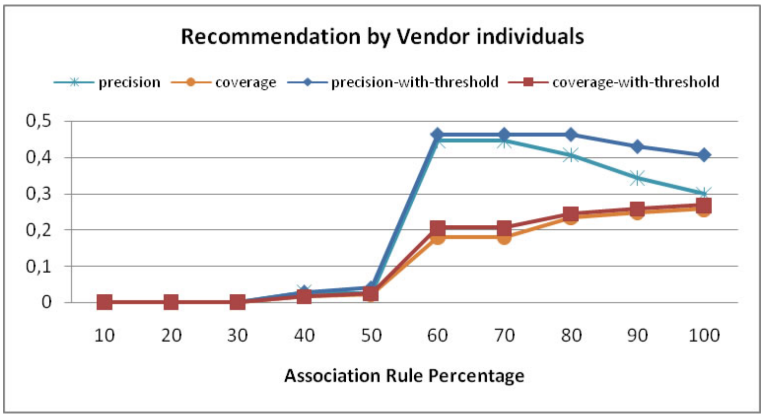

Figure M.3: Vendor class, Minimum Support Threshold: 3% , SECKIN Web Site

APPENDIX N

EFFECT OF NUMBER OF ASSOCIATION RULES ON RECOMMENDATION QUALITY, CENG WEB SITE

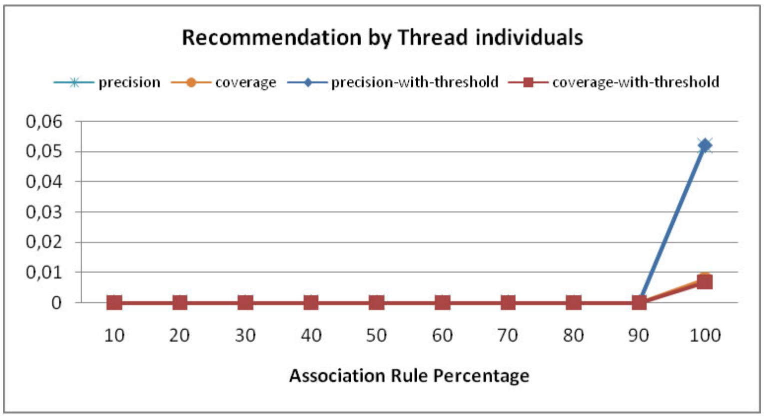

Figure N.1: Thread class, Minimum Support Threshold: 2% , CENG Web Site

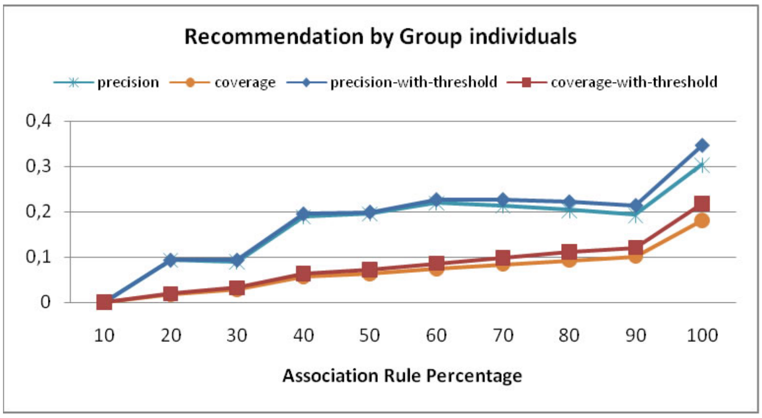

Figure N.2: Group class, Minimum Support Threshold: 8% , CENG Web Site

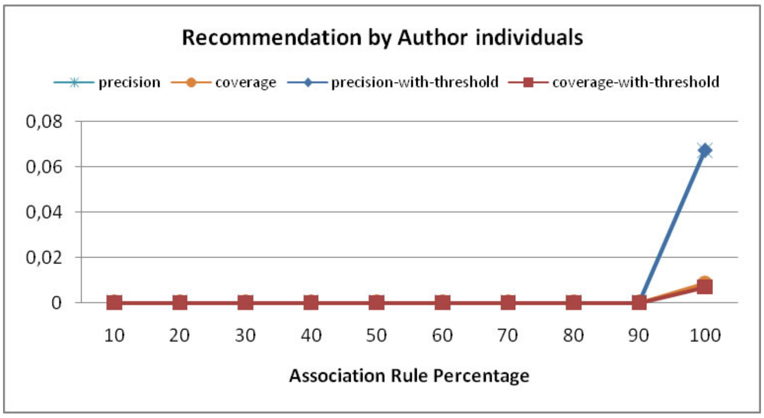

Figure N.3: Author class, Minimum Support Threshold: 2% , CENG Web Site

APPENDIX O

EFFECT OF NUMBER OF ASSOCIATION RULES ON TIME PERFORMANCE, SECKIN WEB SITE

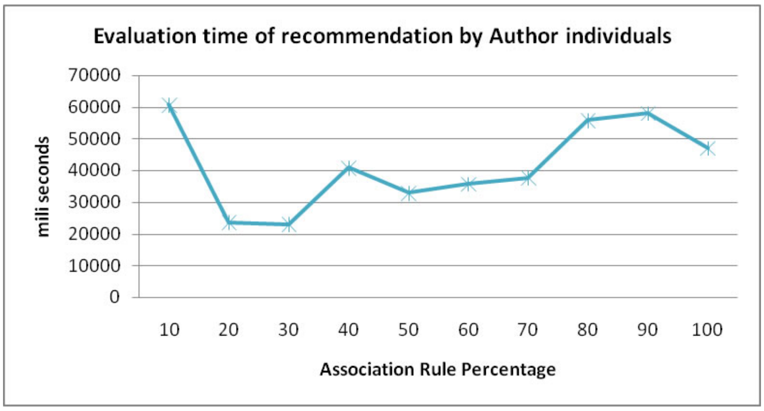

Figure O.1: Author class, SECKIN Web Site

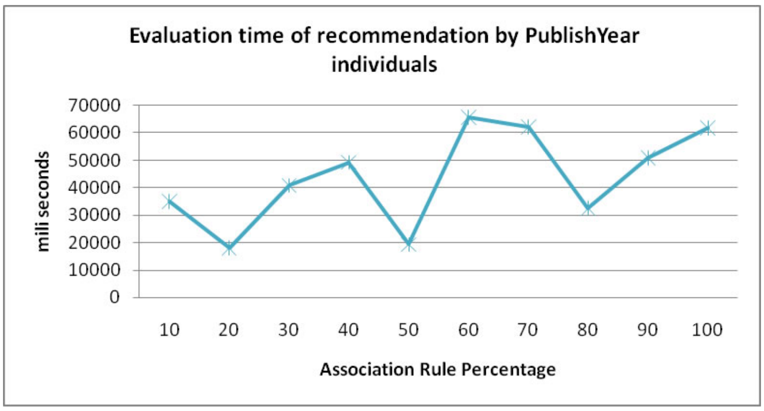

Figure O.2: PublishYear class, SECKIN Web Site

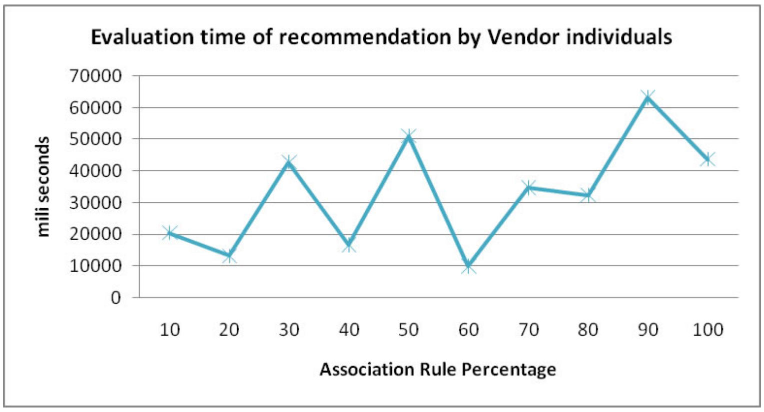

Figure O.3: Vendor class, SECKIN Web Site

APPENDIX P

PERMISSION TO USE THEIR WEB SITE VISIT LOG FROM SECKIN PUBLICATION

seçkin

ORTADOĞU TEKNİK ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜNE

Orta Doğu Teknik Üniversitesi Bilgisayar Mühendisliği Master Programı öğrencisi Süleyman SALIN akademik çalışmalarında, **www.seckin.com.tr** adresindeki web sitemizin 2008 yılına ait ziyaretçi verilerini, aşağıdaki taahhütler dahilinde kullanabilir.

- Web sitemiz ziyaretçilerine ait kişisel hiçbir bilgiyi, hiç bir şekilde akademik çalışmalarında yayınlamamayı ve üçüncü kişi-kurumlara vermemeyi taahhüt eder.
- Kişi bilgileri içermeyecek şekilde verilerin istatistikî ve analiz işlemlerinde kullanabilir; tezinde, bu bilgileri veya analiz sonuçlarını yayınlayabilir.
- Bu izin sadece 2008 yılı www.seckin.com.tr sitesine ait ziyaretçi bilgilerinin ilgili kişinin akademik tez çalışmasında kullanımı çerçevesi ile sınırlıdır.

Seçkin Yayıncılık San. Tic. A.Ş.
Adına

Koray SEÇKİN

SEÇKİN YAYINCILIK
SANAYİ VE TİCARET A.Ş.
Sağlık Sokak No: 21 06410 Sıhhiye - Ankara
Tel: (0.312) 435 30 30 Faks: (0.312) 435 24 72
ÇANKAYA V.D. - NO: 758 009 4271
seçkin

Seçkin Yayıncılık A.Ş.
Merkez : Sağlık Sokak No: 19/B - 21 06410 Sıhhiye - Ankara Tel: (0 312) 435 30 30 Faks: (0 312) 435 24 72 www.seckin.com.tr E-Posta: seckin@seckin.com.tr
Şube : Abide-i Hürriyet Caddesi Arzu Pasajı No: 229/9 Şişli - İstanbul Tel: (0 212) 234 34 77 Faks: (0 212) 231 24 69

Figure P.1: Permission from SECKIN Publication