A NEW REACTIVE METHOD FOR PROCESSING WEB USAGE DATA

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

BY

MURAT ALİ BAYIR

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

JUNE 2006

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. İsmail Hakkı Toroslu
Supervisor

Examining Committee Members

Prof. Dr. Adnan Yazıcı	(METU, CENG)	_____
Assoc. Prof. Dr. İsmail Hakkı Toroslu	(METU, CENG)	_____
Assoc. Prof. Dr. Ahmet Coşar	(METU, CENG)	_____
Asst. Prof. Pınar Şenkul	(METU, CENG)	_____
Güven Fidan (M.S.)	(AGMLab Inc.)	_____

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: Murat Ali Bayır

Signature:

ABSTRACT

A NEW REACTIVE METHOD FOR PROCESSING
WEB USAGE DATA

BAYIR, Murat Ali

MS, Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. İsmail Hakkı Toroslu

June 2006, 82 pages

In this thesis, a new reactive session reconstruction method 'Smart-SRA' is introduced. Web usage mining is a type of web mining, which exploits data mining techniques to discover valuable information from navigations of Web users. As in classical data mining, data processing and pattern discovery are the main issues in web usage mining. The first phase of the web usage mining is the data processing phase including session reconstruction. Session reconstruction is the most important task of web usage mining since it directly affects the quality of the extracted frequent patterns at the final step, significantly. Session reconstruction methods can be classified into two categories, namely 'reactive' and 'proactive' with respect to the data source and the data processing time. If the user requests are processed after the server handles them, this technique is called as ‘reactive’, while in ‘proactive’ strategies this processing occurs during the interactive browsing of the web site. Smart-SRA is a reactive session reconstruction techique, which uses web log data and the site topology. In order to compare Smart-SRA with previous reactive methods, a web agent simulator has been developed. Our agent simulator models behavior of web users and generates web user navigations as well as the log data kept by the web server. In this way, the actual user sessions will be known and the successes of different techniques can be compared. In this thesis, it is shown that the sessions generated by Smart-SRA are more accurate than the sessions constructed by previous heuristics.

Keywords: Web Mining, Web Usage Mining, Session Reconstruction, Agent Simulator, Web Topology

ÖZ

WEB KULLANIM VERİLERİNİ İŞLEMEK İÇİN
SONRADAN AKTİF YENİ BİR METOD

BAYIR, Murat Ali
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. İsmail Hakkı Toroslu

Haziran 2006, 82 sayfa

Bu tez, kullanıcı oturumlarının yeniden oluşturulmasını sağlayan yeni bir sonradan aktif metod olan 'Smart-SRA' yi öne sürmektedir. Ağ Kullanım Madenciliği bir Ağ Madenciliği türü olmakla beraber kullanıcıların ağ üzerinden yaptığı gezinti bilgilerine veri madenciliği metodları uygulayarak değerli veriler çıkartır. Veri işleme ve kalıp çıkarma klasik veri madenciliğinde olduğu gibi ağ kullanım madenciliğinde de iki temel unsurdur. Veri işleme fazı kullanıcı oturumlarının yeniden oluşturulmasını içerir ve aynı zamanda veri madenciliğinin ilk fazıdır. Kullanıcı oturumlarının yeniden oluşturulması işlemi Ağ Kullanım Madenciliğinin en önemli fazıdır, çünkü bu faz Ağ Kullanım Madenciliği sonucunda ortaya çıkan sık kalıpların kalitesini önemli derecede etkilemektedir. Kullanıcı oturumlarının yeniden oluşturulması işlemi, kullanılan veri tipi ve işleme zamanına göre 'sonradan aktif' ve 'önceden aktif' olmak üzere ikiye ayrılır. Kullanıcı istemleri sunucu tarafından yakalanıp kaydedildikten sonra işlenirse bu tekniğe 'sonradan aktif', eğer aynı işlem kullanıcının interaktif olarak sayfa görüntülemesi sırasında olursa 'önceden aktif' denir. 'Smart-SRA' sonradan aktif bir metod olup, sunucu tarafından oluşturulan ağ kullanım verilerini ve sitenin ağ yapısını kullanır. Bu tez kapsamında 'Smart-SRA' nın daha önceki 'sonradan aktif' metodlarla karşılaştırılmasını sağlamak amacıyla ağ kullanım ajani simülatörü geliştirilmiştir. Ağ kullanım ajanı simülatörü ağ kullanıcı davranışlarını modelleyip, sunucu tarafındaki ağ erişim kayıtlarıyla aynı anda ağ kullanıcı gezintileri oluşturmaktadır. Bu yöntemle, gerçek kullanıcı oturumları tanımlanabilip, yeni metodun onceki metodlarla doğruluk performansı

karşılaştırılabilmektedir. Bu tez, ‘Smart-SRA’ tarafından yeniden oluşturulan kullanıcı oturumlarının önceki ‘sonradan-aktif’ metodların oluşturduğu oturumlara oranla daha doğru sonuçlar verdiğini göstermektedir.

Anahtar kelimeler: Ağ Madenciliği, Ağ Kullanım Madenciliği, Kullanıcı Oturumlarının Yeniden Oluşturulması, Ajan Simülatörü, Ağ Yapısı.

To my dear father Ali İhsan Bayır and my dear mother Havva Bayır

and,

to my sister Gamze Bayır

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my thesis supervisor Assoc. Prof. Dr. İsmail Hakkı Toroslu for his guidance, advice, criticism, encouragements and insight throughout the research.

I would also like to thank Assoc. Prof. Dr. Ahmet Coşar for his suggestions and comments.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER I

INTRODUCTION

As in classical data mining, the aim in web mining is to discover and retrieve useful and interesting patterns from a large dataset. There has been huge interest towards web mining. In web mining, this dataset is the huge web data. Web data contains different kinds of information, including, web documents data, web structure data, web log data, and user profiles data. Two different approaches are proposed on the definition of web mining. One approach is process-based and the other is data-based. Data-based definition is more widely accepted today. In this perspective, web mining is the application of data mining techniques to extract knowledge from web data, where at least one of structure or usage data is used in the mining process. There are no differences between web mining and data mining compared in general. All of web data can be mined mainly in three different dimensions, which are; web content mining, web structure mining, and web usage mining.

This thesis is mainly related to web usage mining, which is an important branch of web mining. Web usage mining can be defined as the application of data mining techniques to web log data in order to discover user access patterns. Web usage mining has various application areas such as web pre-fetching, link prediction, site reorganization and web personalization. Most important phases of web usage mining are the reconstruction of user sessions by using heuristics techniques and discovering useful patterns from these sessions by using pattern discovery techniques like apriori or similar ones.

The data processing phase for web usage mining can vary with respect to source and processing time. If we are processing requests after they are handled by the web server, this technique is called "reactive" while in "proactive" techniques the same (pre) processing occurs during the interactive browsing of the web site by the user. In proactive strategies, the raw data is collected when web server is processing client requests. Proactive strategies are more appropriate for dynamically created server pages. Also, in proactive strategies, association of an agent with a session is determined during the interaction of user with web site. However, in reactive strategies, the available raw data is mainly server logs containing information about client requests. Reactive strategies are mostly applied on static web pages. Because the content of dynamic web pages changes in time, it is difficult to predict the relationship between web pages and obtain meaningful navigation path patterns.

In this work, a new reactive session reconstruction method is proposed. This new heuristic uses both the web usage data information and the site topology in order to produce more accurate sessions. For evaluating accuracy of the new method, a web agent simulator is implemented. Comparisons of different session reconstruction heuristics are performed by using agent simulator.

This thesis is organized as follows: In chapter two, a general introduction to data mining and more detailed information about web mining are given. In chapter three, Web Usage Mining, which is a subarea of web mining related to this thesis, is presented. In the fourth chapter, previous reactive session reconstruction heuristics and Smart-SRA are explained. In the fifth chapter, agent simulator and performance evaluation of Smart-SRA with comparison to previous heuristics are discussed. Finally, our conclusions are given.

CHAPTER II

DATA AND WEB MINING

2.1 Data Mining

Data mining has an important place in today's world. It becomes an important research area since the amount of data available in most of the applications. This huge amount of data must be processed in order to extract useful information and knowledge, since they are not explicit. The definition of data mining is given in [Han 00] as Data Mining is the process of discovering interesting knowledge from large amount of data.

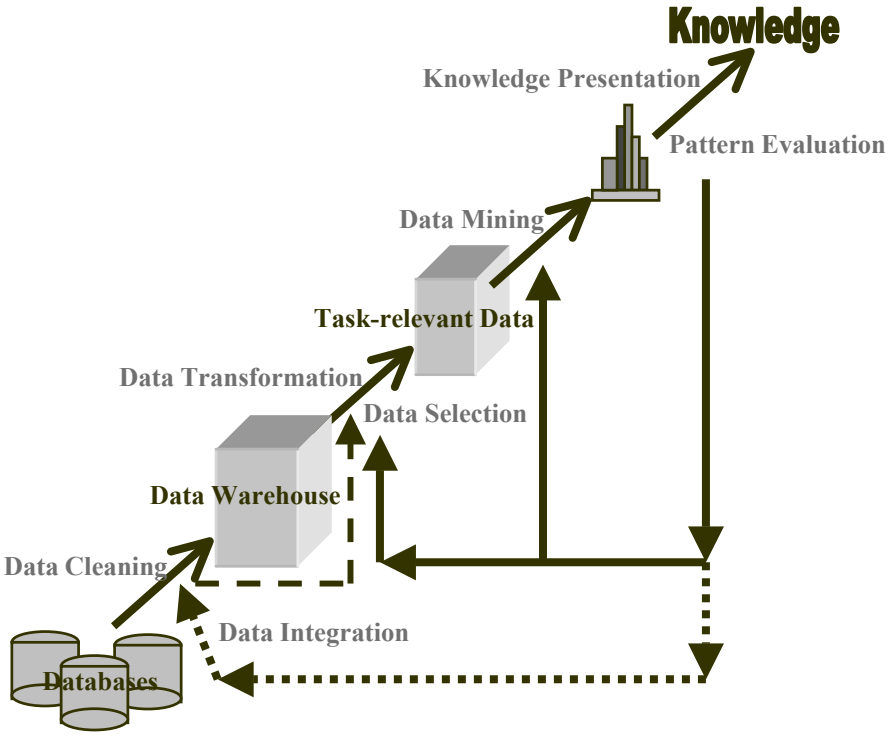

Figure-1 *Phases of Data Mining*

The data source for data mining can be different types of databases such as text files or other types of files including different kinds of data. Data mining is an interdisciplinary research field related to database systems, statistics, machine learning, information retrieval etc. Data Mining is an iterative process consisting the following list of processes:

- data cleaning
- data integration
- data selection
- data transformation
- data mining
- pattern evaluation
- knowledge presentation

The complete data mining process is given in Figure-1.

Data cleaning task handles missing and redundant data in the source file. The real world data can be incomplete, inconsistent and corrupted. In this process, missing values can be filled or removed, noise values are smoothed, outliers are identified and each of these deficiencies are handled by different techniques. Data integration process combines data from various sources. The source data can be multiple distinct databases having different data definitions. In this case, data integration process inserts data into a single coherent data store from these multiple data sources. In the data selection process, the relevant data from data source are retrieved for data mining purposes.

Data transformation process converts source data into proper format for data mining. Data transformation includes basic data management tasks such as smoothing, aggregation, generalization, normalization and attributes construction.

In Data mining process, intelligent methods are applied in order to extract data patterns. Pattern evaluation is the task of discovering interesting patterns among

extracted pattern set. Knowledge representation includes visualization techniques, which are used to interpret discovered knowledge to the user.
Data Mining has various application areas including banking, biology, e-commerce etc. These are most well-known and classical application areas. On the other hand, the new data mining applications include processing spatial data, multimedia data, time-related data and World Wide Web.

World Wide Web is one of the largest and most widely known data source. Today, www contains billions of documents edited by millions of people. The total size of the whole documents can be interpreted in many terabytes. All documents on www are distributed over millions of computers that are connected by telephone lines, optical fibers and radio modems. Www is growing at a very large rate in size of the traffic, the amount of the documents and the complexity of web sites. Due to this trend, the demand for extracting valuable information from this huge amount of data source is increasing everyday. This leads to new area called Web Mining [Etzioni 96], which is the application of data mining techniques to World Wide Web. The next section explains general overview of web mining.

2.2 Web Mining

2.2.1 General Overview of Web Mining

There are several reasons for the emergence of web mining [Han 00]. First of all the World Wide Web is huge and effective source for data mining and data ware housing. The size of the web is very large on the orders of terabytes and it still growing rapidly. Many organizations, individuals or societies provide their public information through web. Also, the content of the web pages are much more complex than any other traditional text documents. Today, web pages lack standard structure, they contain more complex style than standardized formats.

The World Wide Web can be considered as a huge library. Nevertheless, a lot of documents in this huge library are not arranged according to any particular order. Another reason is that the web is a highly dynamic information source. In addition to amazing growth; World Wide Web's information is also updated frequently. News, stocks and markets, company advertisements and Web service centers update their pages regularly. The World Wide Web serves to a broad diversity of user communities. The Internet currently connects about 50 million workstations and the user community of these systems is increasing rapidly. Web users may have different backgrounds, interests and usage purposes. It is also stated that only a small portion of information in the web is relevant or useful. Any web user can be interested in only small portion of the web.

The challenges listed above leads to a research for effective discovery and use of resources in World Wide Web, which also leads to web mining. The Whole schema results in new research area called Web Mining. Indeed, there is no major difference between data mining and web mining. Web Mining can be defined as application of data mining techniques to extract knowledge from the web data including web documents, hyperlinks between documents, usage logs of websites, etc. [Srivastava 02]. Two different approaches were proposed for defining Web mining. The first approach is a 'process-centric view', which defines Web mining as a sequence of ordered tasks [Etzioni 96]. Second one is a 'data-centric view', which defines web mining with respect to the types of web data that was used in the mining process [Cooley 97]. The data-centric definition has become more acceptable [Madria 99, Borges 99, Kosala 00]. Web Mining can be classified with respect to data it is uses. Web involves three types of data [Madria 99]; the actual data on the WWW, the web log data obtained from the users who browsed the web pages and the web structure data. Thus, the web mining should focus on three important dimensions; web structure mining, web content mining and web usage mining. The detailed overview of web mining categories is given in the next.

2.2.2 Types of Web Data

World Wide Web contains various information sources in different formats. As it is stated above World Wide Web involves three types of data, the categorization is given in Figure-2.

Web content data is the data, which web pages are designed for presenting to the users. Web content data consists of free text, semi-structured data like HTML pages and more structured data like automatically generated HTML pages, XML files or data in tables related to web content. Textual, image, audio and video data types falls into this category.

The most common web content data is HTML pages in the web. HTML (Hypertext Markup Language) is designed to determine the logical organizations of documents with hypertext extensions. HTML was firstly implemented by Tim Berners-Lee at CERN, and became popular by the Mosaic browser developed at NCSA. In 1990s it has become widespread with the growth of the Web. After that, HTML has been extended in various ways. The www depends on the web page authors and vendors sharing the same conventions of HTML. Different browsers in various formats can view an HTML document in different ways. To illustrate, one browser may indent the beginning of a paragraph, while another may only leave a blank line. However, base structure remains the same and the organization of document is constant. HTML instructions divide the text of a web page into sub blocks called elements. The HTML elements can be examined in two categorizes: those that define how the body of the document is to be displayed by the browser, and those that define the information about the document, such as the title or relationships to other documents.

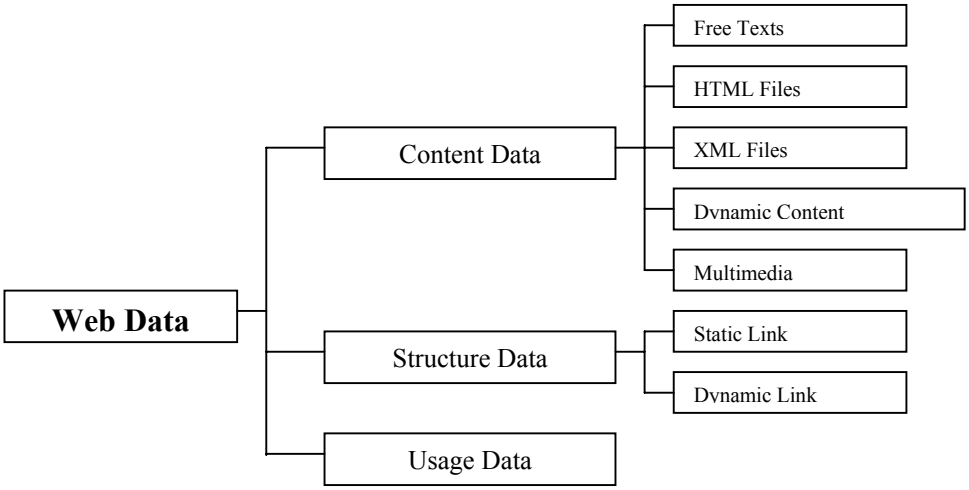

Figure-2 *Types of Web Data*

Another common web content data is the XML documents. XML is a markup language for documents containing structured information. Structured information contains both the content and the information about what content includes and stands for. Almost all documents have some structure. XML has been accepted as a markup language, which is a mechanism to identify structures in a document. XML specification determines a standard way to add markup to documents. XML doesn't specify semantic or tag set. In fact it is a meta-language for describing markups. It provides mechanism to define tags and the structural relationships. All of the semantics of an XML document will either be defined by the applications that process them or by style sheets.

Dynamic server pages are also important part of web content data. Dynamic content can be any web content, which is processed or compiled by the web server before sending the results to the web browser. On the other hand, static content is content, which is sent to the browser without modification. Common forms of dynamic content are Active Server Pages (ASP), Pre-Hypertext Processor (PHP) pages and Java Server Pages (JSP). Today, several web servers support more than one type of active server pages.

Web structure data describes the organization of the content. Intra-page structure information includes the arrangement of various HTML or XML tags within a given page. Inter-page structure information is hyper-links connecting one page to another. Web graph is constructed by hyperlinks information from web pages. The web graph has been widely adopted as the core describing the web structure. It is most widely accepted way of representing web structure related to web page connectivity (dynamic and static links).

The Web graph is a representation of the WWW at a given time [Gandhi 04]. It stores the link structure and connectivity between the HTML documents in the www. Each node in the graph corresponds to a unique web page or a document. An edge represents an HTML link from one page to another. The general properties of web graphs are given below:

- Directed, very large and sparse.
- Highly dynamic
  – Nodes and edges are added/deleted very often
  – Content of the existing nodes are also subject to change
  – Pages and hyperlinks created on the fly
- Apart from primary connected component there are also smaller disconnected components

The size of the web graph is varying from one domain to another domain. An example we graph of a particular web domain is given in the Figure-3.

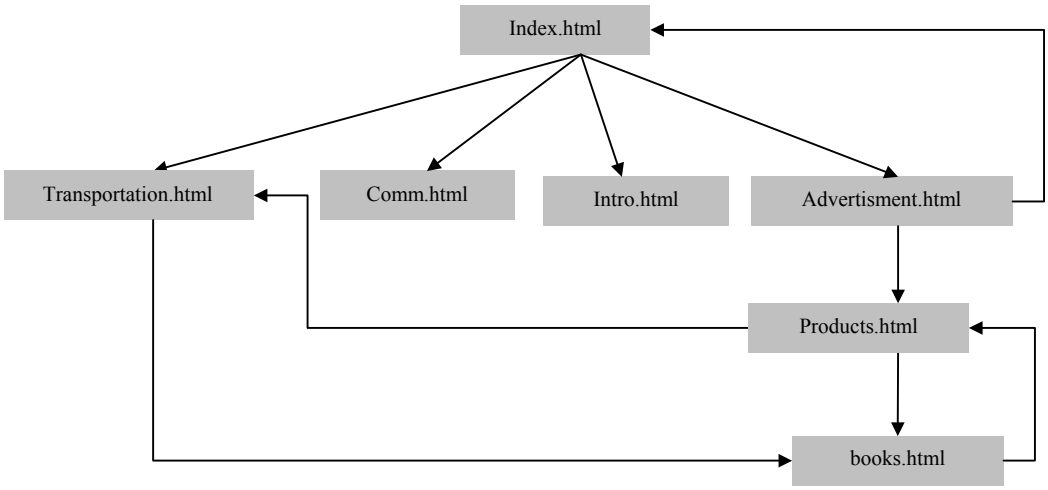

Figure-3 *An Example Web Graph for a Particular Web Domain*

The edges of web graph has the following semantics: Outgoing arcs stands for hypertext links contained in the corresponding page and incoming arcs represent the hypertext links through which the corresponding page is reached. Web graph is used in applications such as web indexing, detection of web communities and web searching. The whole web graph grows with an amazing rate. More specifically, in July 2000, it was estimated that whole web graph consists of about 2.1 billions nodes [Murray 02] and 15 billions edges, since the average node has roughly seven hypertext links (directed edges) to other pages [Kleinberg 99]. In addition, approximately 7.3 millions nodes are added every day and many nodes are modified or removed, so that the Web graph might currently contains more than 7 billions nodes and about 50 billions edges in all.

Web usage data includes web log data from web server access logs, proxy server logs, browser logs, registration data, cookies and any other data generated as the results of web user interactions with web servers. Web log data is created on web server. Every Web server has a unique IP address and a domain name. When any user enters (a URL) in any browser, this request is send to the web server. After that

operation, web server fetches the page and sends it to user's browser. Web server data are created from the relationship between web user's interaction with a web site and the web server. A web server log, containing Web server data, is created as a result of the httpd process that is run on Web servers [Buchner 98]. All types of server activities such as success, errors, and lack of response are logged into a server log file [Bertot 97]. Web servers dynamically produce and update four types of "usage" log files: access log, agent log, error log, and referrer log.

Web Access Logs has fields containing web server data, including the date, time, user's IP address, user action, request method and requested data. Error Logs includes data about specific events such as "file not found," "document contains no data," or configuration errors; providing server administrator information on "problematic and erroneous" links on the server. Other type of data recorded to the error log is aborted transmissions. Agent logs provides data about the browser, browser version, and operating system of the requesting user.

Generally, Web server logs are stored in Common Logfile Format [CLF] or Extended Logfile Format. Common Logfile Format includes date, time, client IP, remote log name of a user, bytes transferred, server name, requested URL, and http status code returned. Extended Logfile Format includes bytes sent and received, server name, IP address, port, request query, requested service name, time elapsed for transaction to complete, version of transfer protocol used, user agent which is the browser program making the request, cookie ID, and referrer. Web server logging tools, also known as Web traffic analyzers, analyze the log files of a Web server and produce reports from this information from this data source. These data can be used in the planning and optimizing web site structure.

In this section, common forms of web data is explained; the next section summarizes categories of web mining.

2.2.3 Web Mining Categories

Web Mining can be broadly divided into three categories according to the kinds of data to be mined [Srivastava 02]:

- Web Content Mining
- Web Structure Mining
- Web Usage Mining

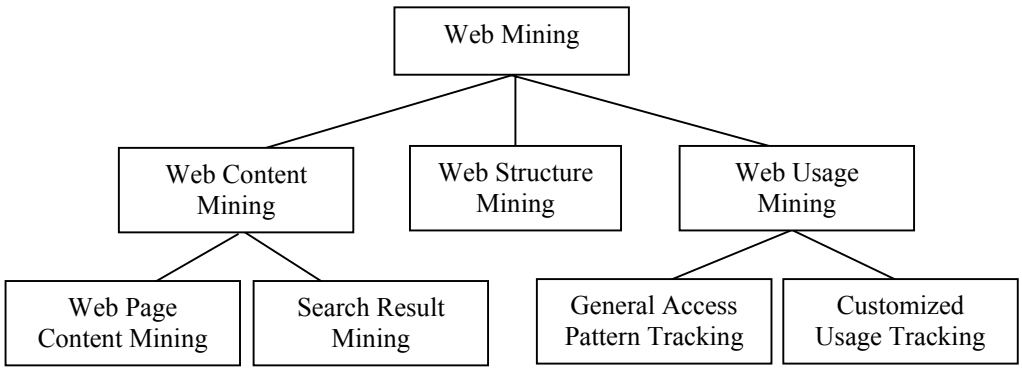

Figure-4 *Taxonomy of Web Mining*

Web content mining is the task of extracting knowledge from the content of documents on World Wide Web like mining the content of html files. Web document text mining, resource discovery based on concepts indexing or agent-based technology fall in this category. Web structure mining is the process of extracting knowledge from the link structure of the World Wide Web. Finally, web usage mining, also known as Web Log Mining, is the process of discovering interesting patterns from web access logs on servers. The Taxonomy of Web Mining is given in Figure-4.

2.2.3.1 Web Content Mining

Web Content Mining is the process of extracting useful information from the contents of Web documents. Content data is the collection of information designed to be conveyed to the users. It may consist of text, images, audio, video, or structured records such as lists and tables. Text mining and its application to Web content has been the most widely studied forms of web content mining. Some of the research issues including the text mining are; topic discovery, extracting association patterns, clustering of web documents and classification of Web Pages. Research activities in these fields also involve using techniques from other disciplines such as Information Retrieval (IR) and Natural Language Processing (NLP). There is also significant body of work exist for discovering knowledge from images in the fields of image processing and computer vision. The application of these techniques to Web content mining has not been very effective yet.

The research on unstructured documents like pure text files also falls into web content mining. Unstructured documents are free texts in www such as newspaper pages. Most of the researches in this area uses bag of words in order to represent unstructured documents [Salton 83]. Other researches in web content mining includes Latent Semantic Indexing (LSI) [Deerwester 90] which tries to transform original document vectors to a lower dimensional space by analyzing structure of elements in the document pile. Another important information resource used in web content mining is the positions of words [Ahonen 98, Cohen 95] in the document. There are also researches focusing on this topic for solving document categorization problem. The use of text compression is also new research area for text classification task. Web content mining applications range from text classification or text categorization to finding extracting pattern or rules. Topic detection and tracking problems are also research areas related to web content mining. Text Mining methods with their document representations included in web content mining are given in Table-1 below [Kosala 00].

Table-1 *Text Mining Method Included in Web Content Mining*

Document Representation	Method
Bag of Words	Episode Rules TFIDF Naive Bayes Bayes Nets Support Vector Machines Hidden Markov Models Maximum Entropy Rule Based System Boosted Decision Trees Neural Networks Logistic Regression Clustering Algorithms K-nearest Neighbor Decision Trees
Bag of Words with n-grams	Self Organizing Maps Unsupervised Hierarchical Clustering Decision Trees Statistical Analysis
Word positions	Episode Rules
Relational	Propositional Rule Based Systems
Phrases	Decision Trees Naive Bayes Bayes Nets Support Vector Machines Rule Based System Clustering Algorithms K-nearest Neighbor

	Decision Trees
Concept Categories	Relative Entropy
Terms	Association Rules
Hyponyms and synonyms	Rule Based System
Sentences and clauses	Rule Learning
Named Entity	Text Compression

2.2.3.2 Web Structure Mining

As it is stated above the web graph composed of web pages as nodes and hyperlinks as edges, which represents the connection between two web pages. Web structure mining can be defined as a task of discovering structure information from the web. The aim of web structure mining is to produce structural information about the web site and its web pages. Unlike Web content mining, which mainly concentrates on the information of single document, web structure mining tries to discover the link structures of the hyperlinks between documents. By using topology information of hyperlinks, web structure mining can classify the Web pages and produce results such as the similarity and relationship between different Web sites.

Web Structure Mining can be classified into two categories based on the type of structure data used. The structural data for Web structure mining is the link information and document structure. Given a collection of web pages and topology, interesting facts related to page connectivity can be discovered. There has been a detailed study about inter-page relations and hyperlink analysis. [Desikan 02] provides an up-to-date survey. In addition, web document contents can also be represented in a tree-structured format, based on the different HTML and XML tags within the page. Recent studies [Wang 98, Moh 00] have focused on automatically extracting document object model (DOM) structures out of documents.

There are several researches done in the area of web structure mining. Social Network Analysis is one of the most famous ones [Kautz 97, Chakrabarti 00]. With social network analysis, it is possible to discover specific types of pages such as hubs and authorities. The separation is made with respect to the number of incoming and outgoing links. One work [Kautz 97], which is related to social networks analysis, aims to model network of AI researches. Another research in this area is modeling Web topology such as HITS [Kleinberg 98], Page Rank [Brin 98] and improved version of HITS by using content information with link structure. The methods addressed in these works are mainly used to calculate quality and relevance relation of two web pages. Another application of web structure mining is discussed in the context of web Warehouse [Madria 99]. In this work, frequency of local links that is on the same web server is measured. This process helps to measure completeness of the web site. Also measuring replication of documents across web warehouse contributes to find out the mirrored sites.

Web structure mining has another dimension, which is the discovering the structure of Web document. This dimension of web structure mining is used to extract the schema of Web pages. This information is beneficial for navigation purpose and provides comparison and integration of Web page schemes. This type of application of web structure mining provides information to database systems for accessing to web pages by providing a reference schema [Madria 99].

S.Madria et al. [Madria 1999] gave details about how to discover interesting facts describing the connectivity in the Web subset, based on the given collection of connected web documents. The structure information obtained from the Web structure mining has the followings:

- The information about measuring the frequency of the local links in the web tuples in a web table

- The information about measuring the frequency of web tuples in a web table containing links within the same document
- The information measuring the frequency of web tuples in a web table that contains links that are global and the links that point towards different web sites
- The information measuring the frequency of identical web tuples that appear in the web tables.

In general, if a web page is connected to another web page with hyperlinks or the web pages are neighbors, the relationship between these web pages needs to be discovered. The relations between these web pages are categorized with respect to different properties: They may be related by synonyms or ontology, they may have similar contents, they may be on the same server or same person may create them. Another task of web structure mining is to discover the nature of the hierarchy or network of hyperlink in the web sites of a particular domain. This may help to generalize the flow of information in Web sites that may represent some particular domain; therefore the query processing can be performed easier and more efficient. Web structure mining has a strong relation with the web content mining, Because Web documents contain links, and they both use the real or primary data on the Web. It's possible to observe that these two mining areas are applied to same task.

2.2.3.3 Web Usage Mining

Web Usage Mining is the process of applying data mining techniques to discover interesting patterns from Web usage data. Web usage mining provides better understanding for serving the needs of Web-based applications [Sirivastava 00]. Web Usage data keeps information about the identity or origin of Web users with their browsing behaviour in a web domain. Web usage mining itself can be divided into subcategories based on the type of web usage data used.

The fist type of source data for web usage mining is Web Server Data, which stands for the user access logs that are collected at Web server. Some of the typical data collected at a Web server include IP addresses, page references, and access time of the users. For example ‘access.log’ file generated by Apache web server falls in this category. Another type web usage mining source is Application Server Data, which is very common in commercial application servers like Weblogic, BroadVision, and StoryServer. These types of data have significant features in the structural design to enable E-commerce applications to be built on top of them. The importance of these data types comes from their feature to track various types of business events and log them in application server logs.

Application Level Data is another source for web usage mining. With this type of data it is possible to record various kinds of events in an application. These data is used for generating histories about selected special events. The data in this category can be divided into three categories based on the source of its collection: on the server side, the client side, and the proxy side. The important point is that the server side data is an aggregate picture of the usage of a service by all users, while on the client side data is complete picture of usage of all services by a particular client, with the proxy side middle framework. [Sirivastava 00].

Most of the research in Web usage mining is focused on applications using web Server Data. The only remained information after users visits a web site is the data about the path through the pages they have accessed. Most of the Web log analysis tools only use the textual data from log files. They ignore the link information, which is also very important. Web usage mining tries to discover useful information from the data extracted from the interactions of the users while surfing on the Web. It also has focus on the methods predicting user behavior while the user interacts with Web.

The possible application areas of web usage mining are prediction of the user's behavior within the site, comparison between expected and actual Web site usage,

reconstruction of web site structure based on the interests of its users. The detailed information about web usage mining is given in the next chapter.

Many researches have been done in the Database, Information Retrieval, Intelligent Agents and Topology, which provide basic foundation for web content mining, web structure mining. However, web usage mining is a relative new research area, and has more and more attentions in recent years. In the next section, popular application areas of web mining are given.

2.2.4 Web Mining Applications

2.2.4.1 Current Trends in Web Mining Applications

WWW can be accepted as a huge digital library. By the same analogy, Web Mining can be viewed as a digital library's librarian. There are several application areas of web mining; the important ones are listed in Figure-5.

The most popular application area of web mining is e-commerce (business-to-customer) and web based customer relationship management. Web usage mining is most dominant application in this context. With the web mining, it is possible to record customer behaviour for web-based business. It is also feasible to adapt web sites based on interesting patterns as a result of analysis on user navigation patterns [Iyer 02]. Web site topology can be customized to provide better facilities for the site user [Mulvenna 00].

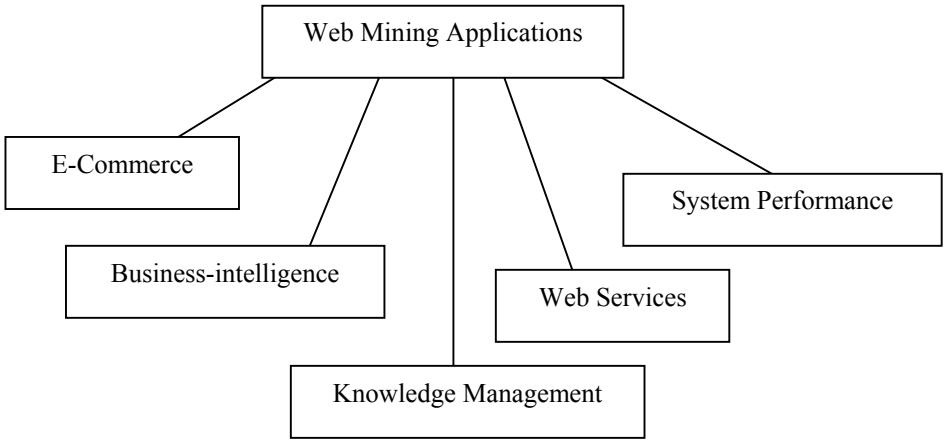

Figure-5 *Application Areas of Web Mining*

Using textual information on the web for predicting customer needs is another research area. [Wuthrich 98, Fung 03] It is possible to associate common phrases with specialized market movements. Knowledge Management Applications can be categorised as “connections” or “collections.” Finding common interest communities like Amazon infers a common interest by identifying those who have bought the same book, and recommending other books that the target community prefers. Membership in forums or news groups is a more direct method of identifying a common interest community.

To sum up, the trend in applications of web mining is on the areas that can profit from business-to-consumer e-commerce, or on the areas where Web-based portals are growing rapidly and providing overlapping, and sometimes conflicting information. In all of these areas, the demand comes from the customer. In e-commerce perspective, Internet shoppers play the most important role. Compared to data mining, web Mining applications depend on the end user for the interpretation of the data presented. For example, fraud detection is a typical data mining application, the end-user expect information from software about potential fraud targets. However, web mining applications tend to fall short of this level of

functionalities. The applications might typically identify linked clusters of textual information from which the end user would identify the important patterns.

2.2.4.2 Recent Applications of Web Mining in Popular Topics

2.2.4.2.1 Robot Detection, Capturing human and non-human Web behavior

Web robot is a software that navigates the hyperlink structure of the World Wide Web for fetching and finding information. The key point for differentiating robot behavior from human behavior and discovering user behavior knowledge from web usage data is addressed in [Kohavi 01]. E-commerce retailers are particularly concerned about the unauthorized deployment of web-robots for finding business intelligence at their Web sites. Also, Web robots tend to consume significant amount of network bandwidth at the expense of other users. Also the session generated by web robots make it more difficult to perform click-stream analysis effectively on the Web data. Identifying the IP address and user agent of the Web clients are two prevalent methods for identifying web-robots. While these techniques are applicable to many well-known robots, they may not be enough to detect previously unknown robots. In [Tan 02], a new approach is proposed which uses the navigational patterns in the click-stream data to determine if it is due to a robot. In addition, it is possible to detect many camouflaging and unknown robots by using this method.

2.2.4.2.2 Extracting User profiles

User profiling issue is taken at many levels in www applications. For instance, in some e-commerce web sites, data collection happens only at the checkout counter, usually called the ‘point-of-sale’. This method gives the data about the result of the whole complex human behavior and decision-making process with no information about the steps of process itself.

If in an e-commerce site, the complete click-stream information is recorded; it gives a detailed record of every single action performed by the user. This also provides much more detailed insight for the decision making process. Adding such behavioral information to other kinds of information about users, e.g. demographic, psychographics, etc. contributes to build more comprehensive user profile, which can be used for many different applications [Masand 02]. Many organizations build web user profiles based on visits to their own sites, Alexa Research [AR] and DoubleClick are successful examples of building www user behaviour profiles. These approaches need browser cookies of some sort, which provides a fairly detailed view of a user's browsing behavior across the Web.

2.2.4.2.3 Finding significant pages in the Web

Hubs and Authorities are accepted two significant nodes in web graph. These two important elements are known as 'fans' and 'centers' in a bipartite core of a Web graph. A Core (i, j) is a complete directed bipartite sub-graph with at least i nodes from Fan and at least j nodes from Center. In the web graph terminology, i pages that contain the links are accepted as 'fans' and the j pages that are referenced are the 'centers'. In web mining context, 'fans' and 'centers' in a Bipartite Core are basically the Hubs and Authorities. The hub and authority coefficients are computed for each Web page, this indicates the extent to which the Web page serves as a "hub" pointing to good "authority" pages or as an "authority" on a topic pointed to by good hubs.

Computing hub and authority scores of single web pages is not based on a single formula. It is calculated by using set of pages relevant to the topic using an step by step approach called HITS algorithm [Kleinberg 98]. The iterative process is given as follows. First a query is submitted to a search engine and relevant documents are fetched. The new set, called the 'root set', then extended by adding Web pages that point to those in the 'root set' and are pointed by those in the 'root set'. The new total set is called the 'Base Set'. An adjacency matrix is formed such that if there exists at

least one hyperlink from page i to page j, then $A_{i,j} = 1$, otherwise $A_{i,j} = 0$. The process continues iteratively until the set does not expand further or a threshold on iterations is reached.

2.2.4.2.4 Search Engine Application, Google

Today, Google [Google-1] is one of the most popular and widely used search engine. Google provides web users with information from more than 2.5 billion web pages that it has indexed on its server. Compared to other search engines, google has a simple and quick search facility. This property makes it the most widely used seach engine. Previous engines based on the web content in order to fetch a web page as a result of submitted query. However, Google was first engine to reflect importance of web structure namely link analysis from the web. The key method of google called PageRank, measures an importance of a page, is the underlying technology in all search products. PageRank method makes use of the information about link structure of the Web graph, this method plays significant role for returning relevant results to simple query.

Web content, especially text and hypertext data is the core data source that google uses. Web graphs contributed to improve goggle's search capability and provide better results for web users. Google improved its search techniques such that it provides customized search to retrieve information for a specific web site. Google use the information about usage statistics. This enhances the quality of Google's results. It provides better search options to reach images and look for pages that have been updated dynamically. Google's web directory provides a fast and easy way to search within a certain topic or related topics.

CHAPTER III

WEB USAGE MINING

Web usage mining is the application of data mining techniques to web click stream data in order to extract usage patterns [Cooley 00]. Recent advances in technology show that the amount of data on the web and complexity of web structure are increasing at a very high rate. Under these conditions, web usage mining plays a very important role in many applications. E-commerce, web personalization, system performance and traffic analysis are common application areas where web usage mining can be used frequently.

Web Usage mining is similar to traditional data mining methods. As in many data mining methods, support and confidence are two key measures used to find the most useful frequent items. If high thresholds are used for web usage mining applications this will often result in a small number of patterns being discovered, which are already obvious in many cases. Similarly, using low thresholds will lead to an explosion in the number of patterns discovered. Therefore, the value of objective measures like frequency threshold should be determined with respect to the application context and the size of the input.

When Application side of web usage mining is analyzed, finding navigation behavior of web users will give valuable information about the amount of time which web users spend on a particular domain like e-commerce portal, business strategies, quality of advertisements and other things. Processing web server logs and customized data for particular web sites such as registration data contributes to better organization of target domain. Reorganizing a web site by using data mining

methods can help to decrease the average navigation path and increase the average page stay time. Also, analyzing requests of web users contributes to clustering web users with respect to different metrics. After this general introduction to the web usage mining, phases of web usage mining are given in the next section.

3.1 Phases of Web Usage Mining

Web Usage Mining consists of three phases as shown in Figure-6 which are named *data processing*, *pattern discovery* and *pattern analysis*. In data processing phase, raw data for performing Web Usage Mining are generated. This phase has two parts called *data cleaning* and *session reconstruction*. Session reconstruction is the most important task in web usage mining since the quality of mined patterns depends on this directly. In the pattern discovery phase, special pattern discovery algorithms are applied on raw data which is output of the data processing phase.

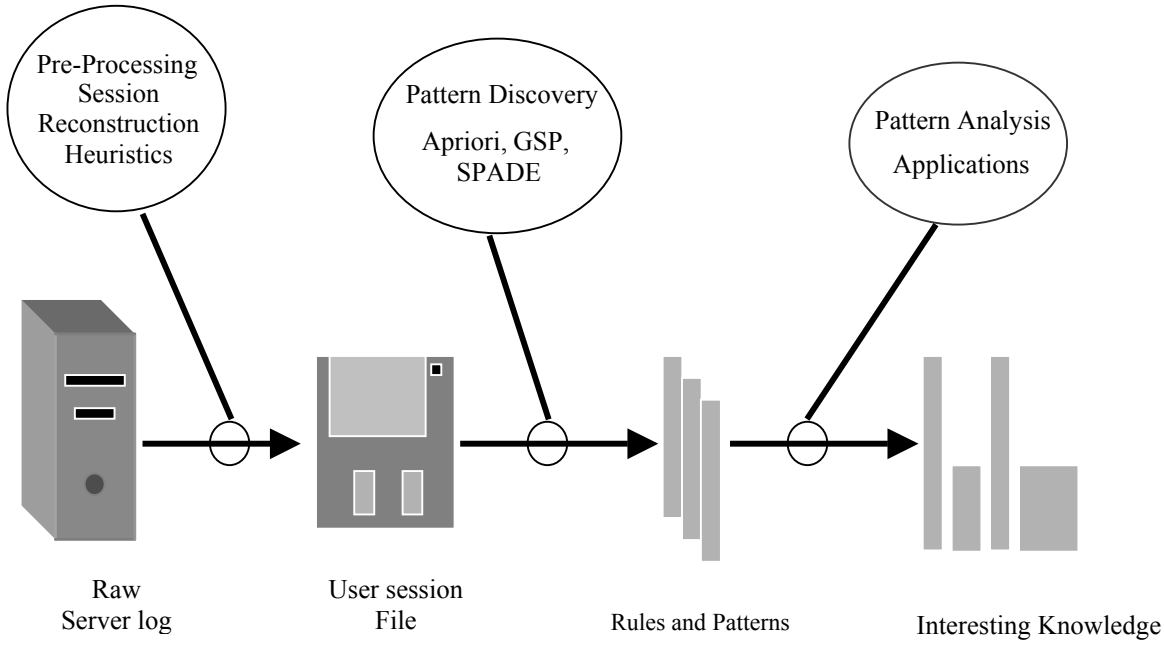

Figure-6 *Phases of Web Usage Mining*

In the pattern analysis phase, interesting knowledge is extracted from frequent patterns and these results are used in various applications such as personalization, system improvement, site modification, e-commerce, etc. The next section discusses the Data processing phase of web usage mining.

3.2 Data Processing Phase of Web Usage Mining

The quality of the patterns discovered in web usage mining process highly depends on the quality of the data used in the mining processes. Web usage data contains information about the Internet addresses of web users with their navigational behaviour. The basic information source for web usage mining itself can be classified in three categories, these are:

Application Server Data: Application server softwares like Web logic, Broad Vision, Story Server used for e-commerce applications, have important properties in their structure. These properties will allow many e-commerce applications to be built on top of them. One of the most important properties of application servers is their ability to keep track of several types of business transactions and record them in application server logs.

Application Level Data: At the application server, the number of event types are increased as we move to upper layers. Application level data can be logged in order to generate histories of specially defined events. This type of data is classified in three categories based on the source of information. These categories are: *server side*, *client side* and *proxy side* data. Server side data gives information about the behaviors of all users, whereas the client side data gives information about a user using that particular client. Proxy side data is somewhere in between the client and server side data.

Web Server Data: This is the most commonly used data type in web usage mining applications. It is data obtained from user logs that are kept by a web server. The basic information source in most of the web usage mining applications is the access log files at server side. When any user agent (e.g., IE, Mozilla, Netscape, etc) hits an URL in a domain, the information related to that operation is recorded in an access log file. In the data processing task, the web log data can be preprocessed in order to obtain session information for all users. In this phase, data reduction techniques can be applied to web log data in order to obtain raw sets for data mining.

Access log file on the server side contains log information of user that opened a session. These logs include the list of items that a user agent has accessed. The log format of the file is Common Log Format [CLF], which includes special record formats. These records have seven common fields, which are:

1. User's IP address
2. Access date and time
3. Request method (GET or POST),
4. URL of the page accessed,
5. Transfer protocol (HTTP 1.0, HTTP 1.1,),
6. Success of return code.
7. Number of bytes transmitted.

The information in this record can be used to recover session information. The attributes that are needed to obtain session information in these tuples are:

1. User's IP address
2. Access date and time
3. URL of the page accessed

In the data cleaning task, redundant fields are removed and also records containing error in any one of these 7 attributes are ignored. The raw data for Web Usage mining has several records containing only the three fields listed above.

Data Cleaning is also a customized step, which includes integrating different usage logs, and parsing data from these usage logs. The requests including redundant URLs are removed from the core data set, for example image file requests are removed during data cleaning. This process can be performed by detecting file types which have suffixes such as *.jpg* and *.png*. The URLs generated by a *spider* or *agent* can be detected and removed during this phase since the amount of requests from this type of sources can be high. This kind of artificial resources will cause error in results when determining navigational behaviors of web users. Also, most of the session reconstruction heuristics only process html files; therefore other types of requests can be eliminated. At the final level, the records including the above 3 key attributes will remain.

After the Data Cleaning task, session reconstruction algorithms are applied to raw data in order to obtain session information for users. The reconstruction of the user activities in sessions for a particular Web site contains a correct mapping of activities to distinct web users and an effective separation of the activities belonging to different sessions of the same individual for that web site. The output of session reconstruction task is also the output of the data processing phase. The details of previous session reconstruction heuristics and our approach is given in the next chapter. The whole session set generated at the end of data processing phase are used as raw data for pattern discovery phase, which is discussed in the next section.

3.3 Pattern Discovery Phase

Pattern discovery is the main issue in both web usage mining and data mining. Many algorithms have been proposed for capturing frequent user navigation patterns.

Finding desired patterns is a very challenging task on very large data. The search space increases exponentially as the lengths of patterns to be discovered increase. Also, discovered patterns must be interpreted and understandable knowledge must be extracted from them. In this section, we give a simple introduction to common algorithms, which can be used in the pattern discovery phase of web usage mining. Also, the comparison of Pattern Discovery on Web Logs Data is given in our work [Bayir 06-1]. Commonly used pattern discovery algorithms that are also suitable for Web Usage Mining are *Apriori*, *GSP* and *SPADE*. In the following sections, these algorithms are introduced.

3.3.1 Apriori Algorithm

Apriori algorithm [Agrawal 94] is an efficient algorithm, which is one of the best available methods for discovering frequent patterns. Apriori is most widely used when working on databases containing transactions. The algorithm serves as a basis for many other pattern discovery methods. Apriori runs breadth-first search algorithm and uses a hash tree structure to count candidate items at each step. Its search is complete and bottom up with a horizontal layout and discovers all frequent *itemsets*.

Apriori is an iterative algorithm that counts *itemsets* of a specific length at each step while going over the database. The initial tasks of the method are scanning all records in the database and finding the first frequent item sets. After this step, using these frequent items, it forms potential frequent candidate 2-itemsets. An additional pass for scanning all transactions in the database is performed for determining supports of these patterns. By observing supports, the infrequent ones are eliminated from candidate 2-itemsets, while the remaining ones will form frequent 2-itemsets. This process is repeated until all frequent itemsets have been discovered. There are three main steps in this algorithm:

1. Generate candidates of length k from the frequent k-1 length item sets, by a self join on Fk-1.

2. Prune any candidate with at least one infrequent subset.

3. Scan all transactions to obtain candidate supports.

The general form Apriori Algorithm is given below:

F_1 = {Frequent-1 itemsets};
For ($k=2$; $F_{k-1} \neq \{\}$; k++)
  C_k = Set of New Candidates;
    For all transactions $t \in D$
      For all k-subsets s of t
        If($s \in C_k$)
          s.count++
    $F_k = \{c \in C_k \mid c.count > min\_sup\}$
Set of all frequent itemsets = $U_k F_k$

An axample application of the Apriori algorithm is given below:

Transaction Data:

Table-2 *Transaction Data for Apriori Example*

TID	Transaction
01	<1 2 3 >
02	<4 3 1>
03	<2 3>
04	<1 2 3 5>

Min_Fre = 0.5, Min_Sup = 2, Min_Fre = (Min_Sup) / # of Transactions = 2 / 4

At the first step, F_1 which is Frequent 1 itemsets is found, the table below gives the whole 1 itemsets with their support, the infrequent items support of which is smaller than the minimum support are eliminated and remaining sets given in Table-4 forms frquent-1 itemsets.

Table-3 *Frequency of All 1 Itemsets for Apriori*

F_1	Frequency
< 1 >	0.75
< 2 >	0.75
< 3 >	1
< 4 >	**0.25 < Min_Fre**
< 5 >	**0.25 < Min_Fre**

Table-4 *Frequent-1 Itemsets for Apriori*

F_1	Frequency
< 1 >	0.75
< 2 >	0.75
< 3 >	1

In the next step, the for loop in the algorithm is executed and Frequent-1 itemsets are joined with each other, where in each join the support of 2-itemset is calculated, after that the frequent itemsets form Frequent-2 itemset (Table-5). The frequent-3 itemsets (Table-6) are discovered similarly. The algorithm stops when the set of frequent-n itemsets becomes empty. In our case the frequent-4 itemsets are given in Table-7.

Table-5 *Frequent-2 Itemsets for Apriori*

F_2	Frequency
< 1 3 >	0.75
< 2 3 >	0.75
< 1 2 >	0.50

Table-6 *Frequent-3 Itemsets for Apriori*

F_3	Frequency
< 1 2 3 >	0.5

Table-7 *Frequent-4 itemsets for Apriori*

F_4	Frequency
Empty	-

3.3.2 GSP Algorithm

GSP [Agrawal 96] is an apriori-based algorithm for solving the sequential pattern-mining problem. The difference between Apriori and GSP comes from candidate generation and support counting. GSP doesn't need separate itemset and sequence phases unlike Apriori. However, it keeps entire sequences in a more complicated tree structure. All combinations of any sequences, with the given identifiers, in the given order are stored in nodes of the hash tree. This type of data structure needs a candidate sequence generation phase that uses itemset and sequence candidate generation phases similar to that in the Apriori. In GSP the sequence length corresponds to number of items, not the number of itemsets. For example <(0 1) (2)> is a sequence with length-2 in Apriori terminology, however, it is a length-3 sequence if we are using GSP terminology.

Similar to the Apriori algorithm, in the first step, GSP finds frequent items at level one. After the first level, at level two; GSP has its own constraints for generating new candidate sequences. The process of candidate generation at each step is similar to Apriori except the elements in the generated new candidate patterns are in the same place in the ancestor pattern. This comes from the sequential pattern criteria. The multiple occurrences of a candidate in any pattern will increase the support of the candidate by only one. However, this property differs in transactional databases. Like the Apriori, GSP also has the disadvantages of generating long candidates and performing many database scans.

To form any sequence with length-n by using GSP, two sequences such that items 2 to (n-1) of the first sequence must be exactly the same as items 1 to (n-2) of the second sequence. To give an example, <(a b) (c d) (e)> and <(b) (c d e f)> meet the first condition. Subsequently, to verify that the length-n sequence could exist, all contiguous length-(n-1) sub-sequences have to extracted before exploring a length-n sequence.

At each step of GSP, frequency criteria for sequence-subsequence relation are used for pruning candidate sequences for the next step. If a sequence is frequent, each of its possible subsequence must also be frequent. As an example, if the sequence <1 2 3> is frequent then all of its subsequences <1>, <2>, <3>, <1 2>, <1 3>, <2 3> must also be frequent. By this logic, it can be stated that if a sequence has at least one infrequent subsequence than, the sequence must be infrequent. Obviously, if a sequence is infrequent, new sessions generated from this sequence will also be infrequent. In GSP, at the fist level we have k sequences and the frequency of all sequences are checked. Then, in the next level, there can be k^2 possible sequences. However, there is no need to check all of the candidate sequences (P_1P_1, P_1P_2, ……P_1P_k, P_2P_1, …… P_2P_k,……..P_kP_1,….., P_kP_k), By looking at frequent sequences at the previous level, the sequences that need to be checked are

determined. At the third level, there are k^3 possible patterns. However, there is no need to check all of them as in the second level. In general, it can be stated that, the number of probabilities we have to check is given by the formula: $\sum_{i=1}^{k} (k^{i} - t_{i-1})$

An example application of GSP is given below:

Assume that our transaction data given below keeps the sequence of items.

Table-8 *Transaction Data for GSP Example*

TID	Transaction
01	<1 2 3 >
02	<4 3 1>
03	<2 3>
04	<1 2 3 5>

Min_Fre = 0.5, Min_Sup = 2,
Min_Fre = (Min_Sup) / # of Transactions = 2 / 4

At the first step, F_1 which is Frequent 1 itemsets is found, the table below gives all of the 1 itemsets with their support. The infrequent items, frequency of which is smaller than the minimum frequency, are eliminated and remaining sets given in Table-10 form the frequent-1 itemsets.

Table-9 *Whole 1Itemsets with Their Frequency for GSP*

Level	Sequence	Frequency	Is it frequent?
1	< 1 >	0.75	Yes
1	< 2 >	0.75	Yes
1	< 3 >	1	Yes

1	< 4 >	0.25	No
1	< 5 >	0.25	No

Table-10 *Frequent-1Itemsets for GSP*

Level	Sequence	Frequency	Is it frequent?
1	< 1 >	0.75	Yes
1	< 2 >	0.75	Yes
1	< 3 >	1	Yes

In the next step, Frequent-1 itemsets are merged with each other with respect to the rule indicated in GSP. In each join the support of the generated 2-itemset is calculated, and the frequent ones with support higher than the given threshold form Frequent-2 itemset (Table-12). The frequent-3 itemsets (Table-13) are discovered in the same way. The algorithm stops when the set of Frequent-n itemsets becomes empty which is Frequent-4 itemsets in our case given in (Table-14).

Table-11 *Frequency of All 2 Itemsets for GSP*

Level	Sequence	Frequency	Is it frequent?
2	< 1 1 >	0	No
2	< 1 2 >	0.50	Yes
2	< 1 3 >	0	No
2	< 2 1 >	0	No
2	< 2 2 >	0	No
2	< 2 3 >	0.75	Yes
2	< 3 1 >	0.25	No
2	< 3 2 >	0	No
2	< 3 3 >	0	No

Table-12 *Frequent-2 Itemsets for GSP*

Level	Sequence	Frequency	Is it frequent?
2	< 1 2 >	0.50	Yes
2	< 2 3 >	0.75	Yes

Table-13 *Frequent-3 Itemsets for GSP*

Level	Sequence	Frequency	Is it frequent?
3	< 1 2 3 >	0.50	Yes

Table-14 *Frequent-4 Itemsets for GSP*

Level	Sequence	Frequency	Is it frequent?
4	-	0	No

3.3.3 SPADE

SPADE is a sequential pattern discovery method proposed in [Zaki 01]. The Algorithm uses Lattice-theoretic approach to partition the search space. Before going into details of SPADE, it is better to look at some basic definitions about lattice.

Definition of Lattice: An algebra (L; $\wedge$;$\vee$) is called a lattice, if L is a nonempty set and $\wedge$ and $\vee$ are binary operations on L. Both $\wedge$ and $\vee$ are idempotent, commutative, associative and they satisfy absorption law.

Idempotent: If A is a set, $A \wedge A = A$, $A \vee A = A$.

Commutative: If A and B are sets, $A \wedge B = B \wedge A$, $A \vee B = B \vee A$,

Associative: If A, B and C are sets

$$A \vee (B \vee C) = (A \vee B) \vee C$$

$$A \wedge (B \wedge C) = (A \wedge B) \wedge C$$

Absorption: If A and B are sets $A \wedge (A \vee B) = A \vee (A \wedge B) = A$.

Atoms: Each single item with length-1 in a transaction database called an atom.

The problems with the GSP Algorithm is that it performs multiple databases scans. GSP has a complex hash structure with poor locality and pattern discovery time by GSP Algorithm increases linearly when the number of data increases. SPADE tries to solve these problems aroused by GSP.

SPADE performs Breath-First and Depth-First search on lattice structure and it is claimed that it has better performance than GSP [Zaki 01]. SPADE uses a different data structure for storing sequences and a cross-reference table storing where each sequence occurs. SPADE uses vertical id-list database.

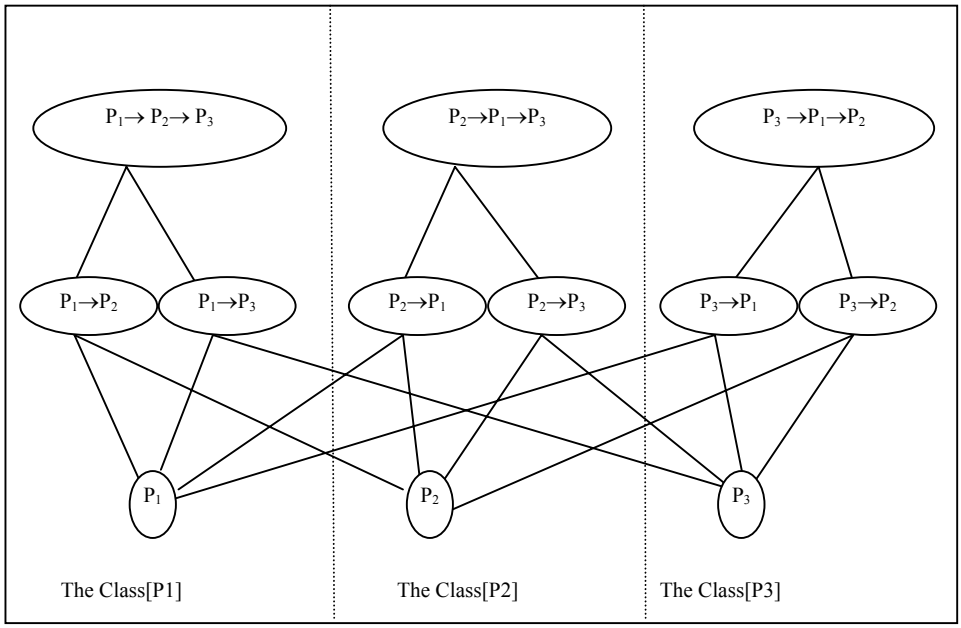

Figure-7 *Example Prefix Based Lattice*

In SPADE, a depth first search starts with length-1 sequences having smaller support. After that, possible sequences that can be generated from the initial sequence are determined. Both BFS and DFS runs on pattern lattice and explore nodes on it. The stoms are at the bottom layers of this lattice structure. An example prefix based lattice (Figure-7) is given below with 3 atoms. The set containing the sequence starting with item P is called class [P].

In SPADE, BFS is more efficient than DFS, because DFS can find any pattern more than once since it explores every sequence starting with a single item. However, BFS will find a sequence only once. Another advantage of the BFS is that previously discovered facts can be used to limit the number of generated candidates, exploring a smaller area in search space and support counting. For reducing the complexity of the problem SPADE stores a list of transaction identifiers where each sequence occurs and provides a support counter. The SPADE algorithm is given below.

SPADE(min_sup,S)

F_1 = all frequent atoms
F_2 = all frequent length-2 sequences
For all equivalance classes $[P_i] \in F_1$ in descending order
$E_2 = [P_i]$
For k=3; $E_{k-1} \neq \{\}$; k++
For All classes $[X] \in E_{k-1}$
N= process_class([X])
If ($N \neq \{\}$)
$E_k = E_k \cup N$
Delete [X]
End For
End For
End For

Process Class function produces all frequent sequences with prefix P_i each join can be performed with the sequence having the same length only in prefix based same equivalence class. The frequency of a new n-sequence generated by joining two n-1 sequences is calculated by using the vertical id-list. There is no need to go over all databases at each step of the algorithm. This leads to significant performance improvement with respect to methods like GSP which uses multiple database-scans. However, usage of vertical-tid lists requires a great amount of memory, and this problem is tried to be solved by processing each equivalence class [X] in the memory independently.

An example application of SPADE is given below:

Table-15 *Original Input-Sequence Database for SPADE*

SID	Time	Items
1	10	CD
1	15	ABC
1	20	ABF
1	25	ACDF
2	15	ABF
2	20	E
3	10	ABF
4	10	DGH
4	20	BF
4	25	AGH

Frequent Sequences with minimum support min_sup=2 given below:

Table-16 *Frequent 1-Sequences for SPADE*

Sequence	Support
A	4
B	4
D	2
F	4

Table-17 *Frequent 2-Sequences for SPADE*

Sequence	Support
AB	3
AF	3
B->A	2
BF	4
D->A	2
D->A	2
D->A	2
F->A	2

Table-18 *Frequent 3-Sequences for SPADE*

Sequence	Support
ABF	3
BF->A	2
D->BF	2
D->B->A	2
D->F->A	2

Table-19 *Frequent 4-Sequences for SPADE*

Sequence	Support
D->BF->A	2

3.4 Applications of Web Usage Mining

Web usage mining has various application areas such as web pre-fetching, link prediction, site reorganization and web personalization. The first two areas are performance issues; the next two application areas can be evaluated in e-business context. Below, you will find example application areas of web usage mining.

3.4.1 Improvement in System Performance

Performance of web services is an important issue for user satisfaction. Web usage mining is an important research area for detecting web traffic behavior, which can be used to develop new policies for increasing the web server performance. Web caching, load balancing, network transmission or data distribution are the common application areas of web mining for performance improvement. Predicting the next web page request for a particular user group is an important research topic in this context [Almeida 96, Menasce 99]. This method is useful especially in web services using static content since dynamic content decrease the usability of web caching at both client and server-level. Using proxy information for pre-fetching a page is also another technique for performance improvement [Aggarwal 97, Cohen 39].

3.4.2 Site Reorganization

The link structure and content structure of any website are two important factors for attractiveness of any web site. The current trend in web mining technologies go towards shorter navigation sequences, for that reason the accessibility of target page in any web domain needs to be increased. Reorganizing site topology of any web domain can achieve this. The reorganization task can be performed with respect to the frequent patterns extracted at the end of web usage mining. Web usage data also gives information about the design of any web site with respect to users behaviors. To illustrate page-stay time gives the pages, which is not attractive. Web site owner can redesign these pages and observe the behavior of users on these pages. These two organizations process both in content and structure leads to adaptive web sites. The example given in [Perkowitz 98, Perkowitz 99] changes web site organization with respect to usage patterns discovered.

3.4.3 Web Personalization

Personalization of web site is one of the key issues in many web-based applications such as individual marketing like electronic commerce. Performing dynamic recommendation is a very important feature in many applications like cross-sale and up sales in e-commerce. Web usage mining is the basic research area, which can be an excellent approach for this type of problems. Existing recommendation systems do not use data mining for recommendations. Web site personalization is based on usage information. Web server logs can be used to cluster web users having similar interests [Yah 96]. This system contains two modules called offline and online. Offline module creates clusters using log information and online module is used for dynamic link generation. Every site user is assigned to single cluster based on his/her current traversal pattern. The links that are showed on browser of any user is the same with other users in the same cluster.

Another project called Site-Helper [Ngu 97] learns user's preferences by looking at accessed page of the target user from web server logs. In this project a list of significant keywords from pages which the user spent significant amount of time is extracted. Based on the frequent keywords from previous pages the recommendation for new pages is made. Web watcher is another project related to web usage mining, it follows user as he/she browses the web and identify the links that are potentially interesting to the users. The web watcher starts with a short description of a user's interests. Each page request is sent through the web watcher proxy, this will easily track the session across multiple web sites and mark any interesting links. Web watcher [Joachims 97] calculates the interestingness of web pages by using user's browsing.

CHAPTER IV

SESSION RECONSTRUCTION HEURISTICS

4.1 Introduction

In this section, previous session reconstruction heuristics and new session reconstruction algorithm proposed in this thesis namely, Smart-SRA (Smart Session Reconstruction Algorithm) is introduced. As it is stated before, the most important phases of web usage mining are the reconstruction of user sessions by using heuristics techniques, and discovering useful patterns from these sessions by using pattern discovery techniques.

Web usage mining data is related to mainly users' navigation on the web. The most common action of the web user is navigation through web pages by using hyperlinks. A web page can be accepted as related to another web page if they are accessed in the same user session; also, similarity increases if both of these pages are accessed in the same navigation of a user. However, since http protocol is stateless and connectionless, it is not easy to discover user sessions from server logs. For reactive strategies, all users behind a proxy server will have the same IP number and will be seen as a single client and all of these users' logs will contain the same IP number in the web log data. Also, caching performed by the clients' browsers and proxy servers make web log data less reliable. These problems can be handled by proactive strategies by using cookies or java applets. However, some clients could have disabled these solutions easily. In this case proactive strategies become unusable. In

previous works on reactive strategies, mainly sessions are reconstructed by using page access timestamps and navigation constraint heuristics [Berendt 03].

The data source for web usage mining can vary with respect to methods used. In proactive strategies [Fu 02, Shahabi 03], the raw data is collected during run time, which is usually supported by dynamic server pages. Also, in these strategies, association of an agent with a session is determined during the interaction of user with web site. However, in reactive strategies [Spiliopoulou 98, Cooley 99-1, Cooley 99-2], raw data is mainly server logs containing information about client requests. In this thesis, only reactive strategies is considered because mining a huge collection of access data captured by web server can be more convenient after the interaction since it doesn't generate extra load on the web server during the interaction. Comparison of reactive approaches with proactive ones is not meaningful either because of different input sets they use. Since web server logs are used for reactive purposes, raw data has the same advantages and disadvantages for reactive heuristics.

When a user agent (Internet Explorer, Mozilla, Netscape, etc) hit an URL in some domain, the information related to that operation is recorded in access log file. Basically an access log file contains its information in Common Log Format [CLF]. An example access log is given in Table-20.

Table-20 *An Example User Web Page Access Log*

IP Address	Request Time	Method	URL	Protocol	Success of Return Code	Number of Bytes Transmitted
144.123.121	[25/Apr/2005:03:04:41–05]	GET	A.html	HTTP/1.0	200	3290
144.123.121	[25/Apr/2005:03:04:43–05]	GET	B.html	HTTP/1.0	200	2050
144.123.121	[25/Apr/2005:03:04:48–05]	GET	F.html	HTTP/1.0	200	4130

In common log format, each user request to any URL corresponds to a record in access log file. As it is mentioned in previous chapter, each record is a tuple containing 7 attributes.

For the session reconstruction, IP address, request time, and URL are the only information needed from the user web access log in order to obtain users' navigation paths. Therefore, other attributes from the log data are ignored.

Reactive strategies are mostly applied to static web pages. Because the content of active web pages changes dynamically, it is difficult to predict the relationship between web pages and obtain meaningful navigation path patterns.

As mentioned above, previous reactive strategies [Berendt 03] for session reconstruction use two types of heuristics. In time-oriented heuristics [Spiliopoulou 98, Cooley 99-1], session data is reconstructed by using the session duration time or time between consecutive web page requests (page stay time). In navigation-oriented approach [Cooley 99-1, Cooley 99-2], session reconstruction is based on hyperlinks among the pages user requested. This heuristic contains browser movements providing path completion. In this thesis, a novel approach is proposed, which combines time-oriented and navigation oriented approaches in order to obtain more accurate sessions. As in navigation-oriented heuristic, our technique also uses the web site topology and includes path completion with a different method. It is a reactive strategy designed for discovering user session patterns on static web pages. Because we assume static web, the target web site to be mined can easily be modeled as a web graph [Andrei 00, Cooper 01, Kumar 00]. The adjacency matrix of this graph represents the relationships among the web pages. In this thesis, we compare our heuristics with all three previously studied reactive strategies. We don't perform any comparison with proactive strategies as they would definitely use more information (e.g. cookies) instead of using only the web log data. Chapter 5 gives the details about our experiments.

The next section summarizes previously used reactive strategies, namely time and navigation oriented heuristics. Then we introduce our heuristic 'Smart-SRA' for session reconstruction.

4.2 Previous Reactive Heuristics for Session Reconstruction

4.2.1 Time-oriented heuristics

Time oriented heuristics [Spiliopoulou 98, Cooley 99-1] are based on time limitations on total session time or page-stay time. There are two types of time-oriented heuristics. In the first one, the duration of session cannot be greater than a predefined upper bound δ. The upper bound δ is usually accepted as 30 minutes according to [Catladge 95]. Any page requested with timestamp t_i can be appended to the current session if the time difference between the requested page's timestamp and the timestamp of the first page t_0 of that session is smaller than δ ($t_i - t_0 \leq \delta$). The first page with time stamp greater than $t_0 + \delta$ becomes the first page of the new session. In other words, if [WP_1, WP_2, …. , WP_N] are web pages forming a session (in increasing order of access time), then access_time(WP_N) – access_time(WP_1) ≤ δ. The pseudocode for time-oriented heuristics using total time constraints given below:

Input:
  L : The set of input logs, |L| : the number of input logs
  δ : user defined upper bound
Output:
  FinalSessionSet

FinalSessionSet = {}
Function Log_Parser_TimeOrineted_I (|L|,L, δ)
  For each L_i of L

```
If Method_i is 'GET' AND Url_i is 'WEBPAGE'
    If ∃ S_k ∈ FinalSessionSet with IP_k = IP_i then
        If ( (Time_i – START_TIME(S_k) ) < δ)  then
            S_k = (IP_k , PAGES_k ● Url_i)
        Else
            Close_session(S_k)
            Open_session(IP_i, Url_i)
        End if
    Else
        Open_session(IP_i, Url_i)
    End if
End if
End For
```

Consider the following table that shows a sample web page requests sequence of an agent in timestamp order.

Table-21 *Example Web Page Request Sequence for First Time-Oriented Heuristic*

Page	P_1	P_{20}	P_{13}	P_{49}	P_{34}	P_{23}
Timestamp	0	6	15	29	32	47

By using a δ value of 30 minutes, we obtain two sessions from the web page request sequence in table-21; the first session is [P_1, P_{20}, P_{13}, P_{49}] and second one is [P_{34}, P_{23}].

In the second time-oriented heuristic, the time spent on any page is limited with a threshold of δ. This threshold value is accepted as 10 minutes according to [Catladge 95]. If t_j is the timestamp of the most recently accessed page P_J, and t_{J-1} is the timestamp of page P_{J-1} accessed immediately before page P_J, then, $t_J – t_{J-1} \leq \delta$ must be

satisfied. Otherwise this new request becomes the first page of the new session. In other words, if [WP_1, WP_2, …, WP_K, WP_{K+1}, … ,WP_N] are pages forming a session, then, $1 \leq K < N$ access_time(WP_{K+1}) − access_time(WP_K) ≤ δ. The pseudocode for time-oriented heuristics using total time constraints given below:

Input:

L : The set of input logs

|L| : the number of input logs

δ : user defined upper bound

Output:

FinalSessionSet

```
FinalSessionSet = {}
Function Log_Parser_TimeOrineted_I ( |L|,L, δ)
   For each Li of L
        If Methodi is 'GET' AND Urli is 'WEBPAGE'
            If ∃ Sk ∈ FinalSessionSet with IPk = IPi then
                If ( (Timei – END_TIME(Sk) ) < δ)  then
                    Sk = (IPk , PAGESk ● Urli)
                Else
                    Close_session(Sk)
                    Open_session(IPi, Urli)
                End if
            Else
                Open_session(IPi, Urli)
            End if
        End if
  End For
```

Consider the following table that shows a sample web page requests sequence of an agent in timestamp order.

Table-22 *Example Web Page Request Sequence for Second Time-Oriented Heuristic*

Page	P_1	P_{20}	P_{13}	P_{49}	P_{34}	P_{23}	P_{15}
Timestamp	0	6	15	29	32	47	54

By using δ as 10 min, we obtain three sessions from the web page request stream above; these sessions are [P_1, P_{20}, P_{13}], [P_{49} P_{34}], and [P_{23}, P_{15}].

In time-oriented approaches, it is very challenging to mine session data correctly, since they do not consider the web page connectivity. In real life, most of the web users request a web page from another one having a hyperlink to it. Also a web page referring to another page can be accepted as related. Thus, it is better to group these pages in the same session. On the other hand, it is better to put two pages into two different sessions if the first one accessed does not have any link to the next one even though the second one is accessed immediately after the first one. Most probably these pages will be unrelated to each other.

4.2.2 Navigation-oriented heuristics

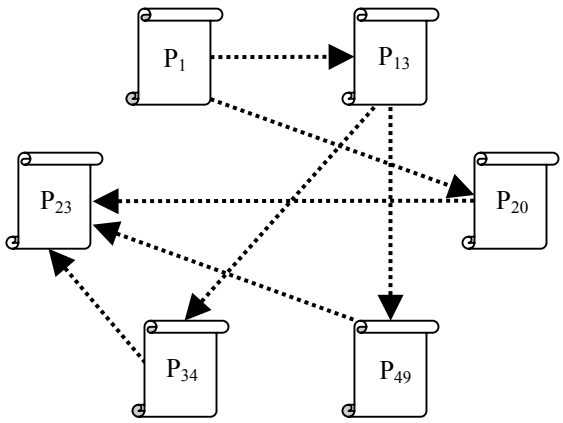

Figure-8 *Example Web Topology Represented as a Graph*

Navigation-oriented approach [Cooley 99-1, Cooley 99-2] uses web topology graph [Andrei 00, Cooper 01, Kumar 00] such as in (Figure-8) corresponding to hyperlinks among web pages, in order to discover sessions. Moreover, in a session, it is not necessary to have a hyperlink between every two consecutive web pages. For every page WP_K (except the initial page WP_1) in a session there must be at least one page WP_J with a hyperlink to WP_K in the same session, having a smaller timestamp than WP_K. If there are several pages having hyperlinks to WP_K with smaller timestamps, then, among these pages, the one with the largest timestamp is assumed to be accessing the page WP_K. Therefore, during the session reconstruction, backward movements until this closest page with a hyperlink to WP_K are appended to that session. In other words, if [WP_1, WP_2, …, WP_K, WP_{K+1}, … , WP_N] are pages forming a session, then, $\forall K$ such that $1 < K \leq N$, $\exists J$ such that, access_time(WP_K) > access_time(WP_J) and there exist a hyperlink from WP_J to WP_K. During the construction of a new session, if [WP_1, WP_2, …, WP_K, WP_{K+1}, … , WP_N] is the current session and WP_{N+1} is a new page, then, the page WP_{N+1} can be added to this session as follows:

- If WP_N has a hyperlink to WP_{N+1}, new session becomes [WP_1, WP_2, … , WP_K, WP_{K+1}, … , WP_N, WP_{N+1}] .
- If WP_N does not have a hyperlink to WP_{N+1}, and WP_K is the nearest (with the largest timestamp smaller than the timestamp of WP_{N+1}) page having a hyperlink to WP_{N+1}, then, the new session becomes [WP_1, WP_2, …, WP_K, WP_{K+1}, … , WP_N, **WP_{N-1}, WP_{N-2}, ..., WP_K**, WP_{N+1}]. The subsequence represented in bold is added to represent the backward browser movements from local cache.

Consider the table-23 that shows a sample web page requests sequence of an agent in timestamp order, and Figure-8 representing the web topology graph, where each page is a node and each directed edge represents the hyperlink between two pages.

Table-23 *Example Web Page Request Sequence for Navigation-Oriented Heuristic*

Page	P_1	P_{20}	P_{13}	P_{49}	P_{34}	P_{23}
Timestamp	0	6	11	14	23	25

Evaluation of final session is given below (Table-24). Backward movements from browser are given in bold.

Table-24 *Evaluation of Example Session by Using Navigation-Oriented Heuristic*

Curent Session	**Condition**	**New Page**
[]	–	P_1
[P_1]	Link[P_1, P_{20}] = 1	P_{20}
[P_1, P_{20}]	Link[P_{20}, P_{13}] = 0 Link[P_1, P_{13}] = 1	P_{13}
[P_1, P_{20}, **P_1,** P_{13}]	Link[P_{13}, P_{49}] = 1	P_{49}
[P_1, P_{20}, **P_1,** P_{13}, P_{49}]	Link[P_{49}, P_{34}] = 0 Link[P_{13}, P_{34}] = 1	P_{34}
[P_1, P_{20}, **P_1,** P_{13}, P_{49}, **P_{13},** P_{34}]	Link[P_{34}, P_{23}] =1	P_{23}
[P_1, P_{20}, **P_1,** P_{13}, P_{49}, **P_{13},** P_{34}, P_{23}]	–	-

In navigation-oriented approach artificially inserting backward movements from browser is a major problem, since although the rest of the session always corresponds to forward movements in web topology graph, backward movements represent movements in reverse direction of the edges, and it is difficult to interpret patterns obtained in this manner. Another problem is the length of sessions. Sessions tend to become longer due to backward movements, and if a navigation-oriented heuristic is used without any time limitation, it is possible to obtain very long

sessions. Lastly, discovering useful patterns from longer patterns becomes more difficult and inefficient.

4.3 A new Method for Session Reconstruction: 'Smart-SRA'

As mentioned in the previous section both time and navigation oriented heuristics have some deficiencies. Smart-SRA [Bayir 06-2] namely 'Smart Session Reconstruction Algorithm' is our new approach which proposes solutions to these deficiencies by combining both of these heuristics and by using the site topology in a way that eliminates the need for inserting the backward browser movements of navigation-oriented heuristic.

Our approach is composed of two phases. In the first phase, shorter request sequences are constructed by using overall session duration time and page-stay time criteria. A sub-session constructed by using these two criteria corresponds to a session formed according to the time-oriented heuristic using both limitations. In the second phase, sessions are partitioned into maximal sub-sessions obeying both the time and the referrer rules. That means each session $[P_1, \dots P_i, P_{i+1}, \dots P_n]$ satisfies the following two conditions:

1. Link Rule: Between each consecutive page pair in a session there must be a hyperlink from the first page to the second page ($\forall$ $i \leq n$ there exists at least one hyperlink from P_i to P_{i+1}).

2. Timestamp Ordering Rule: The request time of the first page in each consecutive page pair must be smaller than the request time of the second page ($Timestamp(P_i) < Timestamp(P_{i+1})$), i.e. no backward move, and the access time difference between two consecutive pages is less than a predefined limit.

Notice that, the overall session duration time limit is already guaranteed by the first phase. The second phase adds referrer constraints, while still ensuring the satisfaction of the second time constraint and eliminating the need for inserting backward moves. Algorithm for extracting these sequences is given in Step2 below.

Session Reconstruction Algorithm

Input: Page request sequence of a user, given in timestamp order, and called as *UserRequestSequence*. The web topology graph including only the nodes appearing in *UserRequestSequence*, and represented as the adjacency matrix, called *Link*.

Output: The set of reconstructed sessions.

Step1: Construct candidate sessions from user page request sequence, (*UserRequestSequence*), by using both of the time-oriented heuristics. That means for each candidate session constructed, both the total duration of the whole session and the time spend on a page in a session will be limited with thresholds. The set of sub-sessions constructed in this step is called as *CandidateSessionSet*.

Step2:

```
//Processing each candidate session
ForEach CandidateSession in CandidateSessionSet
    NewSessionSet := {}
    While CandidateSession ≠ []
        TempSessionSet := {}
        // Select pages from CandidateSession such that no page in
        // CandidateSession is connected to it in web structure graph
        TempPageSet := {}
        ForEach Page_i in CandidateSession
            StartPageFlag := TRUE
```

```
    ForEach Page_j  Where j>i in CandidateSession
        If (Link[Page_j, Page_i]=1) AND (TimeDiff(Page_j, Page_i) ≤ 10) Then
            StartPageFlag := FALSE
    If StartPageFlag = TRUE Then
        TempPageSet := TempPageSet U {Page_i}
// Remove the selected pages (set) from the current session (list/sequence)
CandidateSession:= CandidateSession – TempPageSet
If NewSessionSet = {} Then
    ForEach Page_i in TempPageSet
        TempSessionSet := TempSessionSet U {[Page_i]}
Else
    ForEach Page_i in TempPageSet
        ForEach Session_j in NewSessionSet
            // If the last element of current session has a link to current page
            //and satisfies time requirements
            If  (Link[LastElement(Session_j), Page_i] = 1)
            AND (TimeDiff(Page_j, Page_i) ≤ 10) Then
                TempSession := Session_j
                TempSession.mark := UNEXTENDED
                TempSession := TempSession ● Page_i // Append page to
        session
                TempSessionSet := TempSessionSet U {TempSession}
                Session_j.mark := EXTENDED
            EndIf
        EndFor
    EndFor
EndIf
ForEach Session_j in NewSessionSet
    If Session_j.mark ≠ EXTENDED Then // Select un-extended sessions
        TempSessionSet := TempSessionSet U { Session_j}
```

NewSessionSet **:=** TempSessionSet

EndWhile

EndFor

Notice that only maximal sequences are kept through the iterations and thus, there is no redundant session construction. Moreover, if the web topology graph contains vertices corresponding to web pages that do not appear in the candidate session being processed, these vertices and their incident edges must be removed from the graph prior to the execution.

Consider the table-25 that shows a sample web page requests sequence of an agent obtained after the first phase of the algorithm, and Figure-8 representing the web topology graph.

Table-25 *Example Web Page Request Sequence for Smart-SRA*

Page	P_1	P_{20}	P_{13}	P_{49}	P_{34}	P_{23}
Timestamp	0	6	9	12	14	15

The application of the inner loop (while loop) of the second phase of the session reconstruction algorithm is given in table-26.

Table-26 *Evaluation of Example Session by Smart-SRA*

Iteration	1	2	3	4
CandidateSession	[P_1, P_{20}, P_{13}, P_{49}, P_{34}, P_{23}]	[P_{20}, P_{13}, P_{49}, P_{34}, P_{23}]	[P_{49}, P_{34}, P_{23}]	[P_{23}]
NewSessionSet (before)		[P_1]	[P_1,P_{20}] [P_1,P_{13}]	[P_1,P_{13},P_{34}] [P_1, P_{13}, P_{49}] [P_1, P_{20}]
TempPageSet	{P_1}	{P_{20}, P_{13}}	{P_{49}, P_{34}}	{P_{23}}

TempSessionSet	$[P_1]$	$[P_1,P_{20}]$ $[P_1,P_{13}]$	$[P_1,P_{13},P_{34}]$ $[P_1, P_{13}, P_{49}]$	$[P_1, P_{13}, P_{34}, P_{23}]$ $[P_1, P_{13}, P_{49}, P_{23}]$ $[P_1, P_{20}, P_{23}]$
NewSessionSet (after)	$[P_1]$	$[P_1,P_{20}]$ $[P_1,P_{13}]$	$[P_1,P_{13},P_{34}]$ $[P_1, P_{13}, P_{49}]$ $[P_1, P_{20}]$	$[P_1, P_{13}, P_{34}, P_{23}]$ $[P_1, P_{13}, P_{49}, P_{23}]$ $[P_1, P_{20}, P_{23}]$
Explanation	P_1 is the start page.	Both P_{20} and P_{13} are reachable from P_1	Both P_{49} and P_{34} are reachable from P_{13}, but not from P_{20}	P_{23} is reachable from P_{34}, P_{49} and P_{20}

For the above example, our algorithm discovers the following three maximal sessions satisfying two conditions described above:

- $[P_1, P_{13}, P_{34}, P_{23}]$
- $[P_1, P_{13}, P_{49}, P_{23}]$
- $[P_1, P_{20}, P_{23}]$

We claim that these sessions are more accurate than the sessions constructed by the previous heuristics.

The session construction algorithm processes each candidate session generated according to both time restriction heuristics. Candidate sessions are generated at the first phase of our algorithm by processing a given user web page access sequence. Whenever time difference between two page accesses exceeds the 10 minutes threshold, page access sequence is broken between these two pages into two separate session candidates. While processing user page access sequence before detecting a break point according to page-stay limit if the total duration of sequences of page accesses exceeds 30 minutes threshold, the sequence is also split before a page immediately before the limit.

The second phase of our algorithm is a novel approach for constructing more realistic sessions by utilizing the web topology together with the user access sequence. The main idea is to discover the navigation of the user in the web through the hyperlinks between the web pages. Thus, in the constructed sessions between two successive web pages there must be a hyperlink from the first one to the second one. In addition, artificially inserted backward browser movements of the previously developed navigation oriented heuristics must not exist in the reconstructed sessions.

CHAPTER V

AGENT SIMULATOR AND PERFORMANCE EVALUATIONS

5.1 Agent Simulator

It is not possible to use real user navigation data for evaluating and comparing different web user session reconstruction heuristics since all of the actual user requests cannot be captured by processing server side access logs. Especially the sessions containing access requests served from a client's and proxy's local cache cannot be accurately determined. Therefore, an agent simulator has been developed that generates web agent requests simulating an actual web user. The agent simulator first generates a typical random web page topology and then generates a user agent that accesses this domain from its client site and navigates in this domain like a real user. In this way, we will have full knowledge about the sessions beforehand, and later we can use a heuristic to process user access log data to discover the sessions. Then, we evaluate how successful that heuristic was in reconstructing the known sessions. While generating a session, our agent simulator eliminates web user navigations provided via a client's local cache. Since the simulator knows the full navigation history at the client side, it can determine navigation requests that are served by the web server.

Agent simulator will produce an access log file at server side containing requests provided by web server. All of the heuristics discussed in previous sections use this file as their input. Also the sessions discovered by these heuristics generating their outputs are compared with the original complete session file. For example; consider

an agent with complete page sequences of [P_1, P_{13}, P_{34}] and [P_1, P_{20}] generated by the agent simulator, which is the real sessions. However, in the produced web server log this sequence can appear as [P_1, P_{13}, P_{34}, P_{20}] because browser provides the movement from P_{34} to P_1 through P_{13} using its local cache, meaning these last two movement will not be sent to the web server. We execute heuristics on the server side log data and produce candidate session sequences. These candidate sequences are compared with real session sequences in order to determine the accuracy of evaluated heuristics.

An important feature of our agent simulator is its ability to represent dynamic behaviors of a web agent. It simulates four basic behaviors of web user. These behaviors can be used to construct more complex navigation in a single session. These four basic behaviors constructing complex navigations given below:

5.1.1. A Web user can start session with any one of the possible entry pages of a web site: Most of the time a web user can enter a web site from external domains via links or users can directly type in the web page address in their browser. Regardless of the entrance type, in each web site there are starting pages (Figure-9), such as "index.html". These pages can be starting page of many web agents with the high probability.

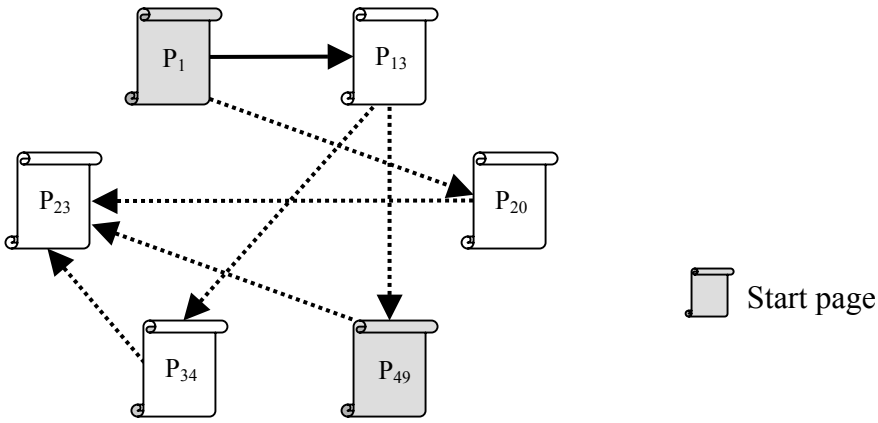

Figure-9 *An Example Web Topology Graph with Given Entry Pages*

For static pages, of course all pages can be typed in address bar and accesses directly. However, not all of the pages will take the first hit from the web users with very high probability. So, most of the pages cannot be accepted as a starting page. While agent simulator creates site topology, it also determines starting pages of the topology. When a user starts a session, the fist page of session is randomly selected from the set of the starting pages. Also, during the navigation, web user can request a new starting page, which cannot be accessible from previous pages. User may type it in web page address. In this case, the new page becomes first page of new session.

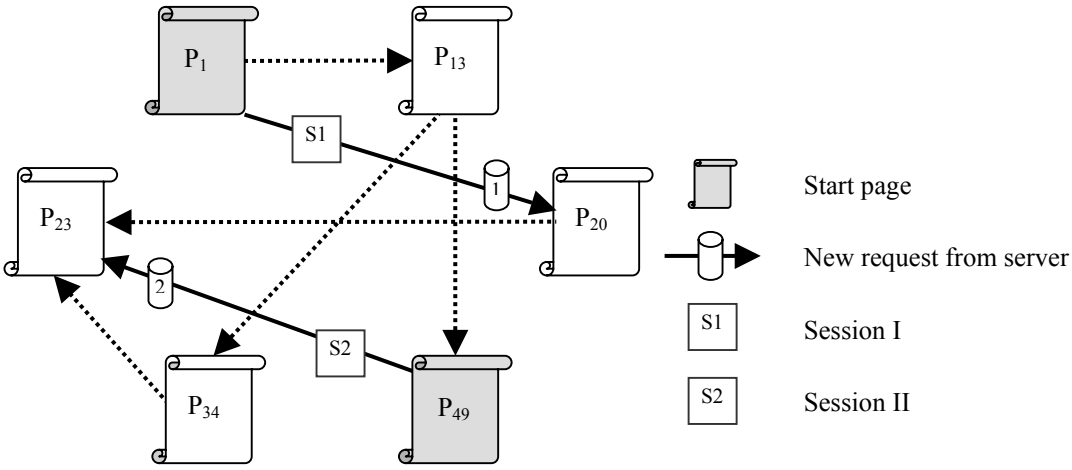

Figure-10 *An Example Behavior of User Requesting Two Entry Pages in Navigation*

We are going to use the example web topology given in Figure-9 in order to illustrate this type of behavior. In this figure, gray pages represent starting pages of the domain. Since P_1 and P_{49} are the starting pages of this topology, the only possible real session list of any agent are in the form of [P_1, …] or [P_{49}, …]. While web user navigating, which had started at some start page, can jump to another start page, such that it is not accessible from previously visited pages. In this case current session terminates and a new session starts. For example, for the navigation of user illustrated in Figure-10, if the current session is [P_1, P_{20}] and user requests P_{49} and P_{23}

consecutively, immediately agent simulator creates a new session [P_{49}, P_{23}] starting with P_{49}, and ends the session [P_1, P_{20}].

5.1.2. A Web user can select the next page having a link from the most recently accessed page: This is the most typical behavior of a web user. When a user is browsing a page, most probably s/he selects one of the links on that page to go to the next one. In order to generate web user navigation, agent simulator first finds pages having links from the current page. Then, one of them is randomly chosen and appended to the end of the current session. This behavior is illustrated in Figure 11. If [P_1, P_{13}] is the current session and the user is browsing page P_{13}, since P_{13} has links to P_{34} and P49, the next page can be one of these two pages. After one of them is selected and a page is appended to the current session, the user navigation sequence becomes [P_1, $P_{13,}$ P_{34}] or [P_1, $P_{13,}$ P_{49}]. In the Figure 11 below web user select P_{34} and session becomes [P_1, $P_{13,}$ P_{34}].

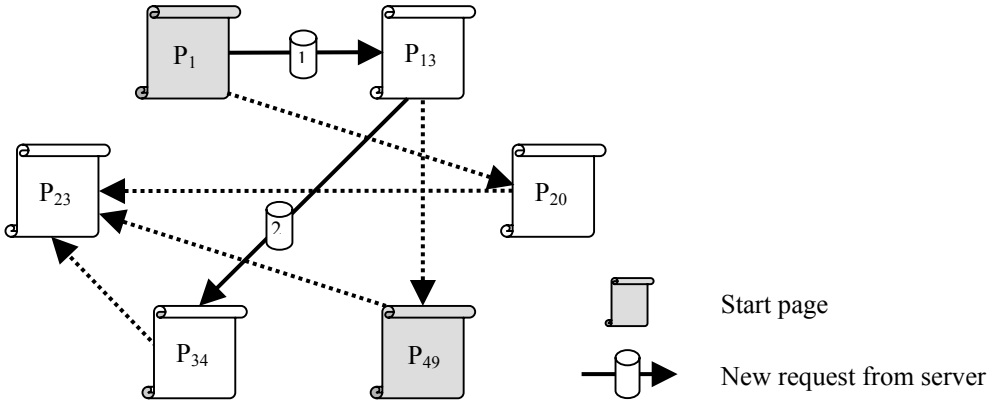

Figure-11 *An Example Navigation of User Requesting Next Pages from Current Page*

5.1.3. A Web user can select as the next page a page having a link from any one of the previously browsed pages (i.e., pages accessed before the most recently accessed one): This behavior is provided by a web browser. Agent simulator

generates browser movements on the client site. However, it also eliminates these movements on the server site while generating log data. By using the web browser, web user can use "back" and "forward" buttons or a link on the current page in order to navigate back to previously browsed page(s) from the local cache, which has been previously accessed. A number of movements towards target page can be provided from browser. Then user can request a new page through web server from target page. In this case, agent simulator selects one of the previously accessed pages that has link to one of the new page not accessed before. For example, if the current session is [P_1, P_{13}, P_{34}], user can return to page P_1 then navigate to P_{20}. In this case the user's requests are: [P_1, P_{13}, P_{34}, **P_{13}, P_1**, P_{20}], bold movements are provided by browser. Agent simulator eliminates such browser movements and, it adds a new session starting from previous page having link to the next page. New real session sequence becomes [P_1, P_{13}, P_{34}] and [P_1, P_{20}]. Notice that agent simulator generated sessions will guarantee that P_i refers P_{i+1}. This type of behavior is illustrated in Figure-12.

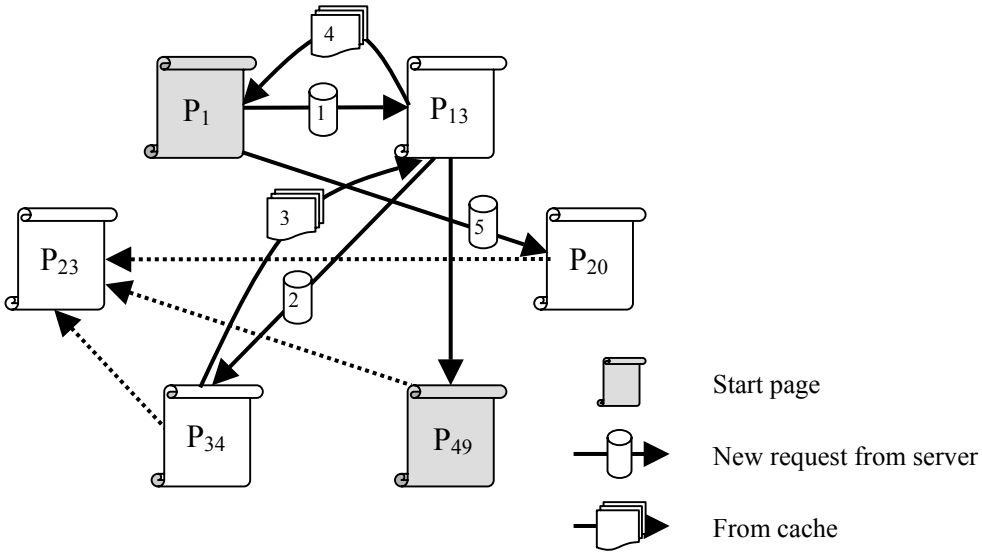

Figure-12 *An Example Navigation of User Requesting Next Pages from Previously Accessed Page*

5.1.4. A Web user can terminate the session: This is a typical behavior. Users can close a browser window, or a time out event could force the session to be terminated invalidating browser links, or user can switch to a new site. In the Figure-13 below user terminates s/he session in P_{23}.

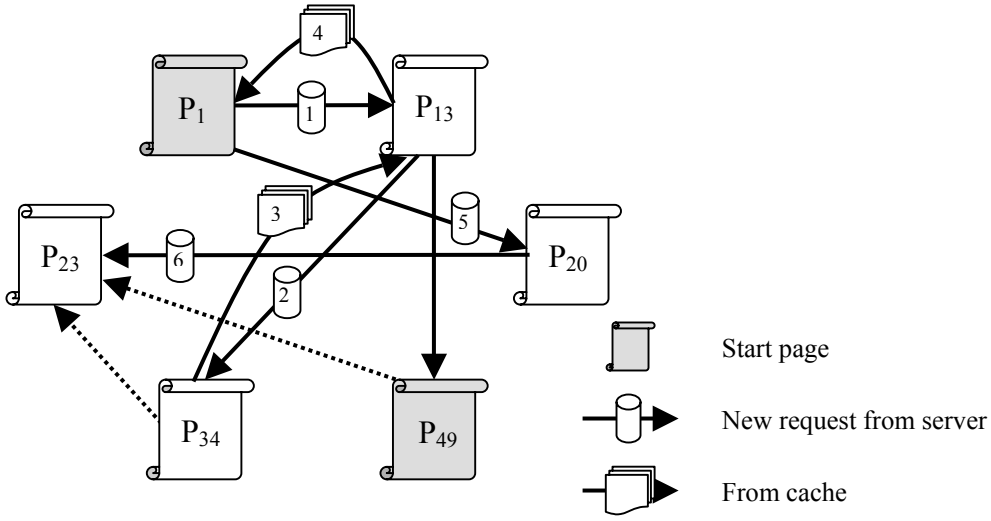

Figure-13 *User Completes Its Navigation in P_{23}*

Agent simulator also uses time considerations while simulating the behaviors described above. In the second and the third behaviors, the time difference between two consecutive page requests is smaller than 10 minutes. Also, in these behaviors, time differences of access time of next page and current page obeys normal distribution. In addition, the median value is taken as 2.12 minutes (from [Berendt 03]), and the standard deviation is taken as 0.5 minutes. The generated time differences set for each type of these behaviors constitute a normal distribution given as in Figure-14.

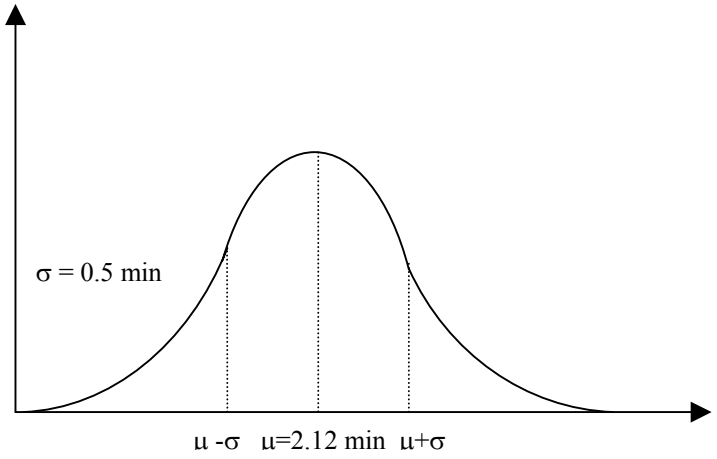

Figure-14 *Normal Distribution Representing Generated Time Difference Sets*

Four primitive basic behaviors given above are implemented in agent simulator. Also following probability parameters are used for simulating navigation behavior of web user.

Session Termination Probability (STP): It is the probability of terminating the current session at any page. The probability of terminating a session should increase as the number of requests of a session increases since it is Poisson probability. The probability of terminating a session until n^{th} request is given as:

Probability of terminating a session until n^{th} request = $(1 - (1\text{-}STP)^n)$.

STP is given between (0,1). For example if STP=0.5 the probability of terminating a session in until 2. Request is $1\text{-}(1\text{-}0.5)^2 = 0.75$, that mean with 0.75 probability the session terminates after 1. Request or 2. Request. In other words with the 0.25 probability the user performs 3. Request.

Link from Previous pages Probability (LPP): LPP is the probability of referring next page from previous pages except the most recently accessed one. This parameter is used to consider backward movements from browser. LPP can be given between [0,1).

New Initial page Probability (NIP): NIP represents the probability of selecting one of the initial pages of web site during the navigation. New Initial page probability (NIP) is provided by the user to control this probability.

After giving basic behaviors of web agent and critical parameters one is remained that Simulating one request of Agent:

```
EndSession := FALSE
NewPage := SelectInitialPage() // Select an initial page
PageSequence := { }
For Each Request While (EndSession = FALSE) do
   CurrentPage :=NewPage
   PageSequence := PageSequence ● CurrentPage  // Append current page
   If (STP> random())
      EndSession := TRUE
   Else If (NIP> random())
      NewPage := SelectInitialPage()  // Always select a new, un-accessed, one
   Else If  (LPP> random())
      CurrentPage := SelectPreviousPage(CurrentPage) // One of the previous pages
      NewPage := SelectPage(CurrentPage)  // Reachable from current page
   Else
      NewPage := SelectPage(CurrentPage) // Reachable from current page
EndWhile
```

5.2 Performance Evaluation

5.2.1 Accuracy Metric

Accurate session must satisfy both the referrer and the time stamp rules explained in previous sections. That is, if S=[S_1, S_2,….. S_i, S_{i+1},….. S_n] is a session containing n pages, then:

- $\forall i < n$ there exists at least one hyperlink from S_i to S_{i+1}
- $\forall i < n$ $T(S_i) < T(S_{i+1})$, Where T(X) is timestamp of page X.

The sessions generated by agent simulator satisfies these two rules. Comparisons of session reconstruction heuristics are performed with respect to 3 parameters, namely STP, LPP and, NIP. The accuracy of a heuristic is defined as the ratio of correctly reconstructed sessions over the number of real sessions generated by the agent simulator as follows:

$$\text{Real Accuracy of Heuristic}_n = \frac{(\#\text{ of session captured by heuristic}_n)}{(\#\text{ of real sessions generated by agent simulator})}$$

In order to evaluate session reconstruction heuristics, first, our agent simulator produces real sessions and proper web log file containing user requests. Then, 4 heuristics use this log file and generates candidate sessions. After that, the accuracies of the heuristics are calculated. For simplicity we use the naming conventions given in Table-27.

Table-27 *Naming Conventions for Session Reconstruction Heuristics*

Heur1	Time oriented Approach (total time ≤ 30 min)
Heur2	Time oriented Approach (page stay ≤ 10 min)
Heur3	Navigation Oriented Approach
Heur4	Smart-SRA

We assume that a session H, reconstructed by a heuristic, captures a real session R, if R occurs as a subsequence of H (represented as $R \subset H$). For example, if R = [P_1, P_3, P_5] and H = [P_9, P_1, P_3, P_5, P_8], then, $R \subset H$ since P_1, P_3 and P_5 are elements of H and they preserve their exact order. On the other hand, if H = [P_1, **P_9**, P_3, P_5, P_8], then, $R \not\subset H$, because P_9 interrupts R in H. Searching real sessions in candidate sessions produced by heuristics can be done by using a simple algorithm adopted from ordinary string searching algorithm.

5.2.2 Experimental Results

For comparisons of different heuristics, using the parameters in Table-28, agent simulator generates random web site topology and web agent navigations. The number of web pages in a web site and the average number of out degrees of the pages (number of links from one page to other pages in the same site) are taken from [SIMS]. Varying values of the three parameters defined in the previous section, namely STP, LPP, and NIP, are used for testing the performances of the heuristics. In our experiments, we have fixed two of these parameters and obtained performance results for the third parameter.

Table-28 *Parameters Used for Generating User Sessions and Web Topology*

Number of web pages (Nodes) in Topology	300
Average number of outdegree	15
Average number of page stay time	2,2 min
Deviation for page stay time	0,5 min
Number of Agents	10000
Fixed Session Termination probability Coefficient (STP)	%5 (0.05)
Fixed Link From previous page Probability (LPP)	%30 (0.3)
Fixed New initial Page probability (NIP)	%30 (0.3)

In the first experiment LPP and NIP are fixed with the values in Table-28, and STP is varying from 1% to 20%. Figure-15 depicts the real accuracy values of 4 heuristics mentioned in this paper for STP values from 1 to 20%. As it is seen from the figure, our heuristic outperforms the other 3 heuristics with a large difference, and the difference is very stable. Our heuristic is almost 1.5-2 times better than the best heuristic for almost all cases.

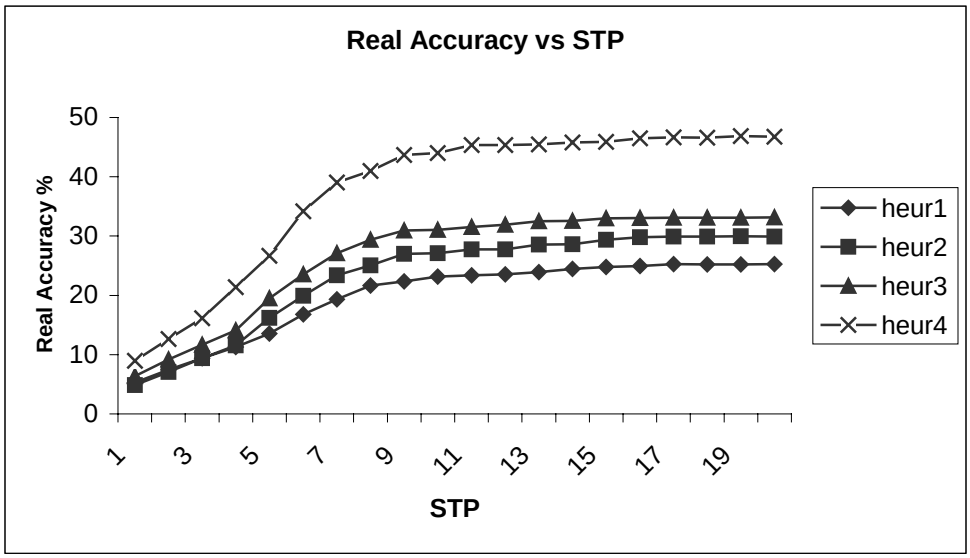

Figure-15 *Real Accuracy Comparison with Increasing STP Value*

Increasing STP leads to sessions with fewer pages. In small sessions the effect of LPP and NIP is also small. For example, in a session with length 2, fixed LPP and NIP values are applied only for the second page. If the navigation is affected by LPP and NIP, then, the session becomes more complex. If there is no return back to an already visited page and there is no new initial page, then, the session becomes simple and it can easily be captured.

In the second experiment, LPP is varying from 0% to 90% and the other two parameters are fixed with the values in Table 9. The results of this experiment are given in Figure-16.

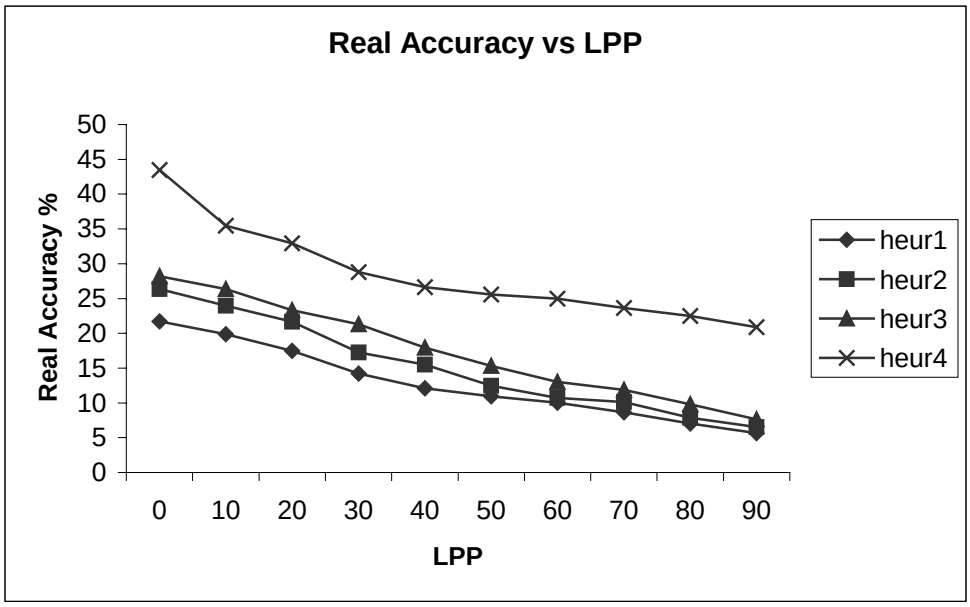

Figure-16 *Real Accuracy Comparison with Increasing LPP Value*

As it is seen from the figure, as LPP increases the real accuracy decreases. Increasing LPP leads to more complex sessions. Path completion is needed for discovering more accurate sessions. Although large LPP values are not very realistic, still we have presented the performance of all 4 heuristics for LPP values up to 90%. For the large LPP values, our heuristic captures nearly 25% of real sessions, whereas other heuristics determine only 6% to 7% of them. Moreover, as in the previous experiment, our heuristic performs at least 1.5-2 times better than the best of the other heuristics for any LPP value.

Similar to previous experiment, in the third experiment, two parameters are fixed with the values in Table 27 and, NIP is varying from 0% to 90%. Performances of 4 heuristics are given in Figure-17.

The results of this experiment are also very similar to the second one. Increasing NIP causes more complex sessions, the accuracy decreases for all heuristics. Also large NIP values is not realistic either. The results of this experiment and the previous one

are also very similar with only one difference; the success of our heuristic is much higher (almost twice as good as the best of the other heuristics) for any NIP value.

As a summary, we can easily say that our new heuristic performs much better than the other three previous heuristics in discovering user sessions for all kinds of parameters.

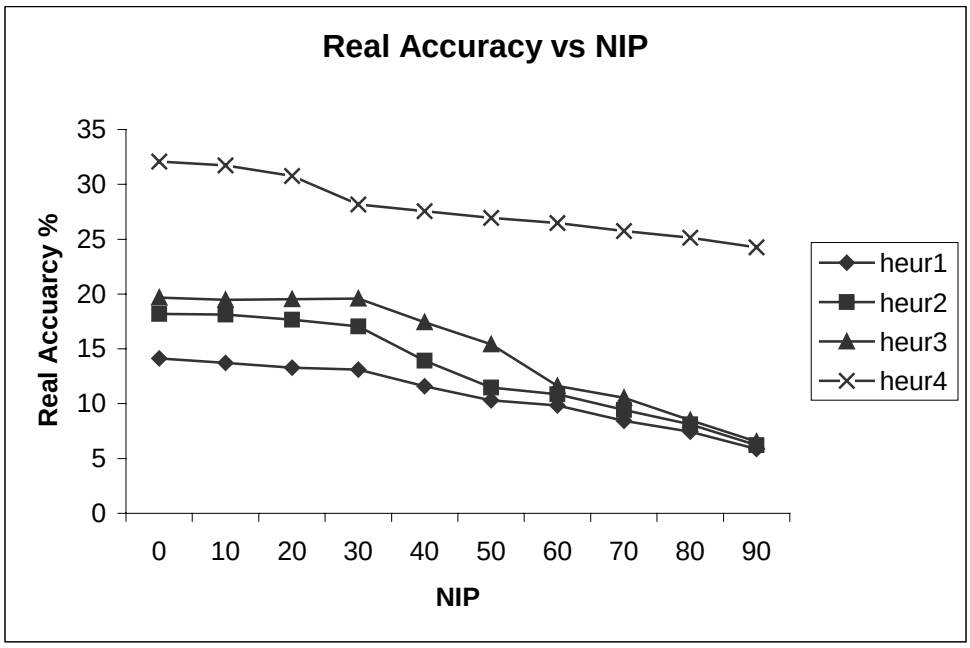

Figure-17 *Real Accuracy Comparison with Increasing NIP Value*

CHAPTER V

CONCLUSION

In this thesis, a new session reconstruction algorithm 'Smart-SRA' is introduced. 'Smart-SRA' is better than previously developed both time and navigation oriented heuristics as it does not allow page sequences with any unrelated (without any hyperlinks from the preceding page to the next page) consecutive requests to be in the same session. Navigation oriented heuristics insert artificial browser (back) requests into a session in order to guarantee that consecutive requests will have connectivity between each other. Thus, Smart-SRA's session sequences are shorter and easier to process than those generated by navigation oriented heuristics. Smart-SRA also enhances navigation-oriented heuristics by using a time oriented extension restricting requests in a session to be at most within a 30 minute period. Another advantage of our heuristic is that it guarantees that all sessions generated will be maximal sequences and do not subsume any other session.

We have implemented agent simulator for generating real user sessions. Our agent simulator generates real sessions satisfying both connectivity and timestamp rules. We have compared the sessions reconstructed by our heuristic and previous heuristics against the real sessions generated by the agent simulator. We have also defined the real accuracy of the constructed sessions as a sequence – subsequence relationship. As it is seen from the experimental results whatever the complex behavior of agent, our approach finds almost 1.5-2 times better solutions than previous heuristics with respect to real accuracy. In addition, it is proved that bigger NIP and LPP values leads to complex navigation behaviors, and thus, intelligent path

completion and separation is needed for guessing more accurate sessions. Larger values of NIP and LPP decrease the accuracy of session reconstruction heuristics. It is shown that our heuristic is also more successful than other heuristics for larger NIP and LPP values. As a result, our approach seems a reasonable method for using reactive web usage mining in real world applications.

REFERENCES

[Agrawal 94] R. Agrawal, R. Srikant (1994), Fast Algorithms for Mining Association Rules in Large Databases. VLDB 94: 487-499

[Agrawal 96] R. Agrawal and R. Srikant (1996), “Mining sequential patterns: Generalizations and performance improvements”, In Proc. 5th Int. Conf. Extending Database Technology (EDBT’96), Avignon, France, pp. 3-17.

[Aggarwal 97] C. C. Aggarwal and P. S. Yu (1997). On disk caching of web objects in proxy servers. In Proc. Int'l. Conf. Info and Knowledge Management, CIKM'97, pages 238 -- 245, Las Vegas, Nevada.

[Ahonen 98] H. Ahonen, O. Heionen, M. Klemettinen and A. Verkamo (1998). Applying data mining techniques for descriptive phrase extraction in digital document collections. In advances in Digital Libraries (ADL 98). Santa Barbara California, USA.

[Almeida 96] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira (1996), "Characterizing Reference Locality in the WWW," In Proceedings of 1996 International Conference on Parallel and Distributed Information Systems (PDIS '96), pp. 92--103.

[Andrei 00] B. Andrei, R. Kumar, M. Farzin (2000), Graph Structure in the Web, Ninth International World wide web Conference, Amsterdam.

[AR] Alexa Research, http://www.alexa.com/

[Bayir 06-1] M. A. Bayir, I. H. Toroslu, A. Cosar, (2006) A Performance Comparison of Pattern Discovery Methods on Web Log Data, AICCSA-06, the 4th ACS/IEEE International Conference on Computer Systems and Applications.

[Bayir 06-2] M. A. Bayir, I. H. Toroslu, A. Cosar, (2006) A New Approach to Reactive Web Usage Data Processing, WIRI06, the 2nd International Workshop on Challenges in Web Information Retrieval and Integration, ICDE 06's Workshop.

[Berendt 03] B. Berendt, B. Mobasher, M. Spiliopoulou, and M. Nakagawa (2003). A Framework for the Evaluation of Session Reconstruction Heuristics in Web Usage Analysis, INFORMS Journal of Computing, Special Issue on Mining Web-Based Data for E-Business Applications Vol. 15, No. 2.

[Bertot 97] J. C. Bertot, C. R. McClure, W. E. Moen & J. Rubin (1997). Web usage statistics: Measurement issues and analytical techniques. Government Information Quarterly, 14 (4), 373-395.

[Borges 99] J. Borges, M. Levene (1998), "Mining Association Rules in Hypertext Databases", in Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), New York City.

[Brin 98] S. Brin and L. Page (1998). The Anatomy of a large-scale hypertextual Web search engine. In seventh international World Wide Web Conference, Brishbane, Australia, 1998.

[Buchner 98] A. G. Buchner, & M. D. Mulvenna (1998). Discovering Internet marketing intelligence through online analytical web usage mining. SIGMOD Record, 27 (4), 54-61.

[Catladge 95] L. Catledge, J. Pitkow (1995), Characterizing browsing behaviors on the world wide web, in Computer Networks and ISDN Systems, 27(6), 1995

[Chakrabarti 00] S. Chakrabarti (2000), Data Mining for hypertext: A tutorial Survey. ACM SIGKDD Explorations. 1(2): 1-11.

[CLF] CERN Common Log Format,
http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format

[Cohen 95] W.W. Cohen (1995). Learning to classify English text with ilp methods. In Advances in Inductive Logic Programming. (Ed. L. De Raed)m IOS Press.

[Cohen 98] E. Cohen, B. Krishnamurthy, and J. Rexford (1998). Improving end-to-end performance of the web using server volumes and proxy filters. In Proc. ACM SIGCOMM, September.

[Cooley 97] R. Cooley, B. Mobasher and J. Srivastava (1997). Web Mining: Information and Pattern Discovery on the Word Wide Web. In Proceedings of the 9th IEEE International Conference on Tools with AI (ICTAI, 97), November.

[Cooley 99-1] R. Cooley, B. Mobasher, and J. Srivastava (1999), Data Preparation for Mining World Wide Web Browsing Patterns . Knowledge and Information Systems Vol. 1, No. 1.

[Cooley 99-2] R. Cooley, P. Tan and J. Srivastava (1999), Discovery of interesting usage patterns from Web data. Advances in Web Usage Analysis and User Profiling. LNAI 1836, Springer, Berlin, Germany. 163-182.

[Cooley 00] R. Cooley (2000) Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data. Ph.D. Thesis. University of Minnesota. May 2000.

[Cooper 01] C. Cooper, A. Frieze (2001), A general model of Web graphs, In ESA, pages 500--511, 2001.

[Deerwester 90] S. Deerwester, S. Dumains, G. Furnas, T. Landauer and R. Harshman (1990). Indexing by Latent Symantic Analysis. Journal of American Society for Information Science. 41(6): 391-407.

[Desikan 02] P. Desikan, J. Srivastava, V. Kumar, P.-N. Tan (2002), “Hyperlink Analysis – Techniques & Applications”, Army High Performance Computing Center Technical Report.

[Etzioni 96] O. Etzioni (1996), “The World Wide Web: Quagmire or Gold Mine”, in Communications of the ACM, 39(11):65-68.

[Fu 02] Y. Fu and M. Shih (2002), A Framework for Personal Web Usage Mining, International Conference on Internet Computing (IC'2002), Las Vegas, NV, pages 595-600.

[Fung 03] G. P. C. Fung , J. X. Yu, et al. (2003). "Stock Prediction: Integrating Text Mining Approach using Real-Time News." IEEE CIFEr'03 Hong Kong: pp.395-401.

[Gandhi 04] M. Gandhi, K. Jeyebalan, J. Kallukalam, A. Rapkin, P. Reilly, N. Widodo (2004), Web Research Infrastructure Project Final Report , Cornell University.

[Google-1] Google Inc. http://www.google.com/

[Google-2] http://www.google.com/press/pressrel/b2b.html

[Han 00] J. Han, M. Kamber (2000): Data Mining: Concepts and Techniques Morgan Kaufmann.

[Iyer 02] G. Iyer , A. Miyazaki (2002), et al. "Linking Web-based segmentation to pricing tactics." Journal of Product & brand Management 11(5): pp.288-302.

[Joachims 97] T. Joachims, D. Freitag, and T. Mitchell (1997). A tour guide for the World Wide Web. In Proceedings of IJCAI97.

[Kautz 97] H. Kautz, B. Selman and M. Shah (1997), The hidden Web, AI Magazine, 18(2),27-36, 1997.

[Kleinberg 98] J. M. Kleinberg (1998), Authoritative Sources in a hyperlinked environment. In Proc-of ACM-SIAM Symposium on Discrete Algorithms, pages 668-677.

[Kleinberg 99] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins (1999). The Web as a Graph: Measurements, Models, and Methods. Proceedings of the International Conference on Combinatorics and Computing, pp. 1-18.

[Kohavi 01] R. Kohavi (2001), “Mining E-Commerce Data: The Good, the Bad, the Ugly”, Invited Industrial presentation at the ACM SIGKDD Conference, San Francisco, CA, http://robotics.stanford.edu/users/ronnyk/kddITrackTalk.pdf .

[Kosala 00] R. Kosala, H. Blockeel (2000), “Web Mining Research: A Survey”, in SIGKDD Explorations 2(1), ACM.

[Kumar 00] R. Kumar, R. Prabhagar, R. Sridhar (2000), The Web As a Graph, 19th ACM SIGACT-SIGMOD-AIGART Symp.

[Madria 99] S. K. Madria, S. S. Bhowmick (1999), W. K. Ng, E. P. Lim: Research Issues in Web Data Mining. DaWaK: 303-312.

[Masand 02] B. Masand, M. Spiliopoulou, J. Srivastava, O. Zaiane (2002), ed. Proceedings of "WebKDD2002 – Web Mining for Usage Patterns and User Profiles", Edmonton, CA.

[Menasce 99] D. Menasce, V. Almeida, R. Fonseca, and M. Mendes (1999), "A methodology for workload characterization of e-commerce sites," in Proc. of ACM Conference on Electronic Commerce (EC-99), Denver, CO, pp. 119--128.

[Moh 00] C. H. Moh, E. P. Lim, W. K. Ng (2000), "DTD-Miner: A Tool for Mining DTD from XML Documents", WECWIS: 144-151.

[Murray 02] B. Murray and A. Moore (2002). Sizing the Internet. White paper, Cyveillance.

[Mulvenna 00] M. Mulvenna , S. Anand, et al. (2000). "Personalization on the Net using Web Mining." Communications of the ACM 43(8).

[Ngu 97] D. Ngu, X. Wu (1997), SiteHelper: A Localized Agent that Helps Incremental Exploration of the World Wide Web, In Proceedings of the Sixth International World Wide Web Conference (WWW6), pp 691-700. Santa Clara, California, USA.

[Perkowitz 98] M. Perkowitz, & O. Etzioni, (1998). Adaptive Web Sites: Automatically Synthesizing Web Pages. In Proceedings of AAAI98.

[Perkowitz 99] M. Perkowitz and O. Etzioni (1999). Adaptive web sites: Conceptual cluster mining. In Sixteenth International Joint Conference on Articial Intelligence, Stockholm, Sweden.

[Salton 83] G. Salton and M. McGill (1983), Introduction to Modern information Retrieval. McGraw Hill.

[Shahabi 03] C. Shahabi, F. B. Kashani (2003): Efficient and Anonymous Web-Usage Mining for Web Personalization. INFORMS Journal on Computing 15(2): 123-147.

[SIMS] http://www.sims.berkeley.edu/
research/projects/how-much-info/internet/ rawdata.html

[Spiliopoulou 98] M. Spiliopoulou, L.C. Faulstich (1998). WUM: A tool for Web Utilization analysis. Proceedings EDBT workshop WebDB'98, LNCS 1590, Springer, Berlin, Germany. 184-203.

[Sirivastava 00] J. Srivastava, R. Cooley, M. Deshpande and P-N. Tan (2000). "Web Usage Mining: Discovery and Applications of usage patterns from Web Data", SIGKDD Explorations, Vol 1, Issue 2.

[Srivastava 02] J. Srivastava, P. Desikan and V. Kumar (2002) "Web Mining: Accomplishments & Future Directions", National Science Foundation Workshop on Next Generation Data Mining (NGDM'02)

[Tan 02] P-N Tan, V. Kumar (2002), Discovery of Web Robot Sessions based on Their Navigational Patterns, Data Mining and Knowledge Discovery, 6(1): 9-35.

[Yah 96] T. Yah, M. Jacobsen, G. Molina, and U. Dayal (1996). From user access patterns to dynamic hypertext linking. In Proceedings of the 5th International World Wide Web Conference, Paxis, France.

[Wang 98] K. Wang and H. Lui (1998), "Discovering Typical Structures of Documents: A Road Map Approach", in Proceedings of the ACM SIGIR Symposium on Information Retrieval.

[Wuthrich 98] B. Wuthrich, D. Permunetilleke, et al. (1998). Daily Prediction of Major Stock Indices from textual WWW Data. 4th International Conference on Knowledge Discovery and Data Mining, pp.364-368.

[Zaki 01] M. J. Zaki (2001), "SPADE: An Efficient Algorithm for Mining Frequent Sequences", in Machine Learning Journal, special issue on Unsupervised Learning (Doug Fisher, ed.), pages 31-60, Vol. 42 No. 1/2.