LINK BASED LIMITED SESSION RECONSTRUCTION METHOD FOR MINING WEB USAGE DATA

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

BURAK TIKNAZ

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

AUGUST 2013

Approval of the thesis:

LINK BASED LIMITED SESSION RECONSTRUCTION METHOD FOR MINING WEB USAGE DATA

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

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

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Prof. Dr. İsmail Hakkı Toroslu _____
Supervisor, **Computer Engineering Department, METU**

Dr. Murat Ali Bayır _____
Co-supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Assoc. Prof. Dr. Ahmet Coşar
Computer Engineering Department, METU _____

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Tolga Can
Computer Engineering Department, METU _____

Assist. Prof. Dr. Aybar Acar
Informatics Institute, METU _____

Dr. Onur Tolga Sehitoglu
Computer Engineering Department, METU _____

Date: _____

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

Name, Last Name: BURAK TIKNAZ

Signature :

ABSTRACT

LINK BASED LIMITED SESSION RECONSTRUCTION METHOD FOR MINING WEB USAGE DATA

Tıknaz, Burak

M.S., Department of Computer Engineering

Supervisor : Prof. Dr. İsmail Hakkı Toroslu

Co-Supervisor : Dr. Murat Ali Bayır

August 2013, 51 pages

Web is growing very fast and serving huge amount of information to people nowadays. Many web users try to access this information every day and for this reason it needs to be organized efficiently. There are traditional web usage mining methods in the literature but detecting user's sessions and understanding their common web behaviors from web logs are difficult problems. In this work, we propose a link based model to session construction problem for finding users' common behaviors on the web. This model aims to find users' sessions and frequent patterns by using web site's topologies and session logs. In order to detect sessions more accurately, we present a new algorithm Limited Session Reconstruction Algorithm. For the pattern discovery phase, an efficient version of Apriori-All technique is used. A web agent simulator is used based on previous works on link based approach to produce web usage logs and site topology. A web tracker tool is designed to capture www.ceng.metu.edu.tr visitor's sessions. Experimental results show that this algorithm gives more accurate results than classical time, navigation approaches and slightly better results than other link based approaches on the simulated data. On the other hand, although it has some enhancements on real data results, its accuracy value is not good compared to some other heuristics due to incompatibility of used web log data and web tracker tool. It is predicted that the new approach gives better results on web sites involving long user sessions such as e-commerce and shopping.

Keywords: Web Usage Mining, Session Reconstruction, Web Topology

ÖZ

LİNKE DAYALI GÜNCEL OTURUM OLUŞTURMA METODU

Tıknaz, Burak
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. İsmail Hakkı Toroslu
Ortak Tez Yöneticisi : Dr. Murat Ali Bayır

Ağustos 2013 , 51 sayfa

Web, son zamanlarda hızla büyümekte ve insanlara büyük miktarda veri sunmaktadır. Birçok web kullanıcısı bu verilere her gün ulaşmaya çalışmakta ve bu yüzden bu verilerin organize edilmesi gerekmektedir. Literatürde birçok web kullanım madenciliği metotları olmakla birlikte web kullanım kayıtlarından kullanıcı oturumlarını tespit etmek ve onların genel davranışlarını belirlemek zor problemlerdir. Bu çalışmada, kullanıcıların web üzerindeki genel davranışlarını bulmak ve oturum oluşturma problemini çözmek için linke dayalı oturum oluşturma modeli öne sürülmüştür. Bu model, kullanıcıların oturumlarını ve sık kullanılan davranışlarını web site topolojilerini ve web kullanım loglarını kullanarak bulmayı amaçlar. Kullanıcı oturumlarının daha doğru tespit edilebilmesi için, yeni bir algoritma olan limitli oturum oluşturma algoritması kullanılmıştır. Sık kullanılan davranış biçimlerinin tespit edilmesi için Appriori-All tekniğinin verimli bir versiyonu kullanılmıştır. Ağ kullanım ajanı modelleyicisi, daha önce bağlantıya dayalı modele göre yapılan çalışmalara dayanarak web kullanım kayıtları ve web topolojileri oluşturulma amacı ile kullanılmıştır. Ağ takipçisi yazılımı www.ceng.metu.edu.tr kullanıcılarının kullanıcı oturumlarını takip etmek amacıyla dizayn edilmiştir. Deney sonuçları, bu algoritmanın geleneksel zaman ve yönelime dayalı algoritmalara göre daha doğru sonuç verdiğini, bununla birlikte daha önceden yapılan bağlantıya dayalı algoritma sonuçlarını biraz daha iyileştirdiğini yapay verilerde göstermiştir. Öte yandan, gerçek verilere göre sonuçlarda iyileştirilmeler olsa da, kullanılan verinin içeriği ve ağ takipçisi yazılımının uygun olmayışı sebebiyle gerçek veri sonuçları karşılaştırılan algoritmaların bir kısmına göre kullanılan web sitesi için iyi değildir. Sonuçların alıveriş ve e-ticaret siteleri gibi uzun kullanıcı oturumları içeren web site verilerinde daha yükseleceği beklenmektedir.

Anahtar Kelimeler: Web Kullanım Madenciliği, Kullanıcı Oturumlarının Yenide Oluşturulması, Web Yapısı

To my family

ACKNOWLEDGMENTS

I would like to thank my supervisor Professor İsmail Hakkı Toroslu for his constant support, guidance, advice and criticism. It was a great honor to work with him for this work. I also would like to thank Dr. Murat Ali Bayır for his support and previous works.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Web Mining consists of data mining techniques to discover patterns from the WEB [17]. Because of the large and growing dataset on the web, it needs to be mined and useful patterns should be retrieved as in data mining. This large dataset can be mined by three different areas, which are web usage mining, web content mining and web structure mining. Web usage mining basically aims to extract useful patterns from web server logs [33]. This is important for many web sites, like e-commerce sites, which serve personalized marketing, since they need good site organization. The process of web usage mining needs user's sessions to detect their behaviors. Reconstruction of user sessions with heuristics and detection of some useful patterns with pattern discovery techniques from these sessions are important parts of web usage mining [10]. This work is related to the Web Usage Mining and aims to detect user's sessions and frequent patterns better.

Session is accepted as a group of search related actions to retrieve information according to [23]. An example session boundary can be known clearly for a user who logins to a web site, then searches content and finally logouts. Since there is no login and logout action generally on the internet while completing this retrieving operation, boundary of a session is not clear [23]. Constructing these sessions by using web usage logs is a harder problem since web logs do not contain state information of users. Thus, reactive session reconstruction heuristics, which use web usage logs, predict this boundary by using time and link informations of web entries.

Sessions can be constructed on the client side by using dynamic server page codes. However, there are some reasons preventing web site owners from inserting these codes into their web sites such as security and keeping internal structure of web sites [8]. Instead of this, mining web usage logs is a good alternative. However, constructing sessions by using user entries in these web logs has two basic problems. The first one is that seperating users is hard because different users may have the same IP address due to proxy servers. The second one is that some of the user requests are not written to the log files due to caching mechanisms of browsers. Reactive strategies try to construct sessions more accurately in spite of these problems. Detecting different users with same IP address is difficult on the web server logs.

On the other hand some of these strategies use web topology graph to solve browser's internal caching problem and produce possible paths on this graph. These heuristics are called link based heuristics and the new proposed heuristic also uses link based approach to construct sessions more accurately.
In this work, we propose a new algorithm based on Link Based Session Model for session construction phase of web usage mining. It uses session logs and web topology graph to reconstruct sessions. We use a web agent simulator to produce data and test results. We implement a program and insert it into our department web site (http://www.ceng.metu.edu.tr) to detect user's sessions for comparing accuracy of algorithms. An efficient version of Apriori-All technique is used to find frequent patterns.

1.2 Organization

This thesis is started with introduction section. Then, detailed information of web mining and web usage mining are given in the second chapter. Chapter two continues to explain web usage mining phases, which are preprocessing, pattern discovery and pattern analysis. Preprocessing part consists of two subsections: cleaning log files and session reconstruction phase. Previous session reconstruction heuristics which are time oriented heuristics, navigation oriented heuristic; Smart Session Reconstruction Algorithm and Complete Session Reconstruction Algorithm are also introduced in section two. In chapter three, the new heuristic Link Based Limited Session Model, agent simulator and web tracker tool are presented. After that, experimental results with simulated data and real data are given in the chapter 4. Finally, this thesis is concluded.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, background knowledge related to this work and related works in the literature are explained. First section contains web mining and its subsections: web content mining, web structure mining and web usage mining. Web usage mining section has also explanations of web usage data, data collection, session management and sequential apriori. The next part continues with introduction of previous approaches for session reconstruction phase. This part includes time oriented heuristics, navigation oriented heuristics, and link based heuristics.

Before explaining web mining, some of the definitions are given here:

A *Uniform Resource Identity* can be explained as a sequence of characters which is used to identify physical or abstract resource according to [11].

A *web server* contains web resources which can be taken by using HTTP protocol.

A *visit* is used to describe sending a request to web server in order to open a web page by using web browser.

A *web browser* is used to display a web page indicated by URI.

A *web page* includes data or resources to display in a web browser.

A *user* is a person who visits web pages by using a web browser.

2.1 Web Mining

Web Mining aims to discover potential useful and unknown information from the web data [25]. Each day, 20 million new web pages are published on the WWW [35]. This growth results in the development of Web Mining. This area consists of three subareas which are: web content mining, web structure mining and web usage mining. These subareas are determined according to the data to be mined. Web content mining mines web documents and resources

[18]. In the web structure mining, used data is web link graph and web usage mining uses web server logs to mine.

2.1.1 Web Content Mining

Web documents and resources that web content mining interested in consist of text, video or audio data in web. This data can be semi structured HTML web pages or structured data which describes self-information: XML, JSON etc. It is different from text mining because of this structured data [20]. Text mining methods only mine unstructured data. Web content mining approaches generally scan massive texts, pictures and graphs to extract results. These results can be used in many domain of it. Some of the popular domains are extracting opinions, reviews and synthesizing knowledge. Search engines have been very popular thanks to web content mining.

2.1.2 Web Structure Mining

Web structure mining uses hyperlinks on the web as mining data. This data consist of nodes, which are web pages, and hyperlinks in these web pages, which are edges [27]. It aims to produce structural information about a web site. After hyperlinks between web pages are analyzed, it will categorize these web pages and extract results such that two different web site's similarity could be extracted by using hyperlinks between web pages.

2.1.3 Web Usage Mining

Web usage mining is the application of using data mining techniques on Web to analyze user access logs. Cooley introduced the term WUM in 1997 [18]. Recently, it has been very popular especially in cross-marketing campaigns to organize web sites according to users and make better advertisements [18].

Web data consists of data provided from web site's usage. Web access log files include these data. This data needs three steps to be mined according to WUM, firstly preprocessing, then pattern discovery and lastly pattern analysis [14].

When a user enters a URL on a web browser on the client side, the server saves the information to the access logs on the server side. These logs should be preprocessed in order to extract useful information about users. The log files have a format, most of them are recorded Common Log Format which is developed by NCSA and CERN [36]. Moreover Combined Log Format, ECLF, is introduced in [21]. A web request from www.ceng.metu.edu.tr's web server log is shown at Figure 2.2. A server web log file consists of these records.

A web log record consists of several parts. Explanations of some of the parts are given in

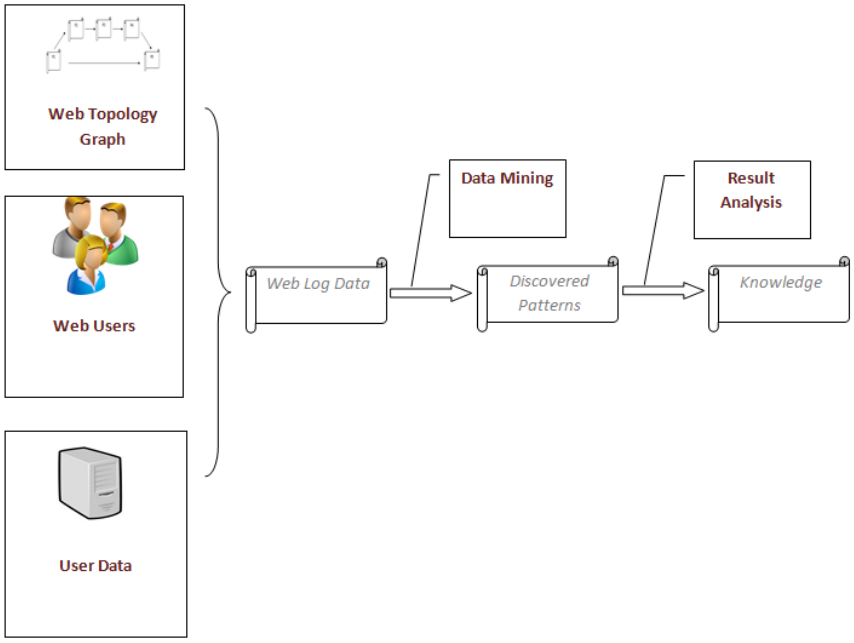

Figure 2.1: Phases of Web Usage Mining

```
88.227.142.123 - - [15/Jul/2012:18:48:14 +0300] "GET /grad/curriculum HTTP/1.1"
200 4070 "http://www.ceng.metu.edu.tr/grad " "Mozilla/4.0 (compatible; MSIE 7.0;
Windows NT 6.1; Trident/5.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729;
.NET CLR 3.0.30729; eSobiSubscriber 2.0.4.16; .NET4.0C; .NET4.0E)"
```

Figure 2.2: A Web request from www.ceng.metu.edu.tr's web server log

Table 2.1.

While processing user data, we need to get user's IP address, the visited page's URL and the access time from the logs to construct user sessions. So, we have to eliminate other items from web logs as well as the entries containing missing fields needed for session reconstruction phase.

2.1.3.1 Preprocessing

2.1.3.1.1 Data Cleaning

Because web logs contain noisy and additional data, this data should be firstly cleaned. Web browsers make implicit requests and generate additional image and script files. Although cleaning methods can be customized, entries including multimedia or image files are gener-

Table2.1: Parts of web request explanations

Web Log Part	**Explanation**
88.227.142.123	Client IP address
[15/Jul/2012:18:48:14 +0300]	Date and time
GET	Type of request (get, post, head...)
/grad/curriculum	Resource URI address
200	Status of the request
4070	Page size
"http://www.ceng.metu.edu.tr/grad"	Referrer page

ally eliminated. These files could be detected according to its suffixes (jpg). Moreover, web robots have noisy sessions in the logs. Because of their high size, they should also be cleaned not to effect data mining technique's success rate.

2.1.3.1.2 Session Reconstruction

The second part of preprocessing phase is session reconstruction. A user session and encountered problems while constructing it were defined in the previous section. Any session consists of the user requests which belong to the same agent and computer. In order to receive session information, session construction algorithms are applied to cleaned data. The reconstruction of sessions separate different web user's actions and a user's different sessions on a web site. Heuristics in the literature generally seperate users by using their IP address. Moreover, URL address and access times are also commonly used fields in the web logs to construct sessions.

Because defining boundary between sessions is impossible by only using web usage log fields, different heuristics focus on different fields to construct sessions more accurately. Differences between web page's access times may give information about a user's behaviour. Sometimes additional information can also be needed. Hyperlink information between web pages is another information preferred by some of the heuristics. Web topology graphs include this information and may give clue about the user's behavior. Some of the heuristics use this information while estimating user's next page request. Heuristics using web topology graph generaly give priority to hyperlink information in order to append a page to an existing session while seperating user's actions to different sessions.

The output of this phase produces the output of data preprocessing. The heuristics about session reconstruction, previous time and navigation based approaches, link based approach [9] and our approach will be discussed in detail in the next chapter.

2.1.3.2 Pattern Discovery Phase

After the data cleaning phase, we have arranged and cleaned logs to discover user patterns. There are many approaches to find user's frequent patterns. In this part, some of these techniques and algorithms are introduced.

2.1.3.2.1 Association Rule Mining

Association Rule Mining algorithms were initially produced to be used in marked-basket analysis [34]. They aim to find web pages that are frequently together in the logs. Here, the term frequency can be determined according to minimum transaction count which is defined as the minimum support. It means that a transaction count of a web page is called frequent if I's value is over minimum transaction count.
A support for association rule can be explained such that: assume that A and B are web page sets, support for this sets is the count of A and B which are occurred together in the logs divided by the total log counts. This value equals to the support for this association rule. On the other hand, confidence value for this set can be calculated by support value of this association rule divided by count of logs including A. This approach gives us more accurate information about A and B relation because it means the probability of visiting page set B when the page set A is visited.
As an example, we have a statement that: "If a www.ceng.metu.edu.tr user visits graduate page, and he also visits curriculum page " has confidence of 75 percent means that entries count including graduate page divided by entries count including both page is 75 percent.
Apriori algorithm [2] is the most known algorithm on this field. This is the most appropriate algorithm to be used with database transactions. Initially, it aims to detect frequent individual items in the database and extend these sets as long as these items' frequencies are enough in the database. It has a hash tree to contain candidate database items and uses breathe first search to make complete search and find all frequent item sets.
This algorithm firstly scans all records in the database to find item sets with sufficient support. After finding records with enough support, then it constructs all 2-itemsets from these item sets. Then algorithm tests whether these data has enough support and eliminate the remaining 2-itemsets. After that, algorithm continues to find 3-itemsets. Until no more frequent item set remains and these datasets cannot grow, these steps are repeated.

Minimum support = 2 (33%), minimum frequency = %33

The last part is finding set of frequent 4-itemsets. Because this set is null, algorithm stops at that point.

Example basically shows that, support counts of items are tested at each iteration and the item sets with enough support value passes to the next iteration. When no new item set with enough support is produced, the algorithm stops.

Table2.2: Appriori Algorithm Example

TID	List of items
T1	l1, l2, l5
T1	l2, l4
T1	l2, l3
T1	l1,l3
T1	l1, l2, l4
T1	l1,l2,l4,l6

Table2.3: The set of frequent 1-itemsets

Item sets	Support count
l1	4
l2	5
l3	2
l4	3
l5	1
l6	1

Table2.4: The set of frequent 1-itemsets with minimum support

Item sets	Support count
l1	4
l2	5
l3	2
l4	3

Table2.5: The set of frequent 2-itemsets

Item sets	Support count
l1, l2	3
l1, l3	1
l1, l4	1
l2, l3	1
l2, l4	3
l3, l4	0

Table2.6: The set of frequent 2-itemsets with minimum support

Item sets	Support count
l1, l2	3
l2, l4	3

Table2.7: The set of frequent 3-itemsets

Item sets	Support count
l1, l2, l4	2

Table2.8: The set of frequent 3-itemsets with minimum support

Item sets	Support count
l1, l2, l4	2

Table2.9: The set of frequent 4-itemsets

Item sets	Support count
Null	-

2.1.3.2.2 Sequential Pattern Mining

Sequential Pattern Mining is similar to association rule mining. It has one more restriction: time. According to sequential pattern mining, the orders of the web pages are important. Our statement is changed and sequential pattern mining aims to find that "If a www.ceng.metu.edu.tr user visits graduate page, then he visits curriculum page". Sequential Pattern mining constructs the support value according to graduate and curriculum page's precise order.

The GSP [1] is one of the most known sequential mining algorithms. Like association mining, it has multiple database passes. It finds 1-sequences in the first step. In the second step, it generates 2-sequences according to its constraints and these steps continue until no more new frequent sequence remains.

Unlike Appriori, a frequent sequence cannot have a subsequence which is not frequent. For example, if a, b, c is a frequent sequence, then a, b, c, a, b, a, c, b, c, a, b, c have to be a frequent sequence.

Table2.10: Gsp Algorithm Example

TID	List of items
T1	l1, l2, l5
T1	l2, l4
T1	l2, l3
T1	l1,l3
T1	l1, l2, l4
T1	l1,l2,l4,l6

Minimum support = 2 (33%), minimum frequency = %33

Table2.11: The set of frequent 1-itemsets

Item sets	**Support count**
l1	4
l2	5
l3	2
l4	3
l5	1
l6	1

Table2.12: The set of frequent 1-itemsets with minimum support

Item sets	**Support count**
l1	4
l2	5
l3	2
l4	3

After first step, we eliminate the non-frequent items. Then we have 4 items. In the second step, we find the possible combinations of each item according to GSP rule, then, the item sets over support count are find.

Table2.13: The set of frequent 2-itemsets

Item sets	**Support count**
l1, l1	0
l1, l2	3
l1, l3	1
l1, l4	0
l2, l1	0
l2, l2	0
l2, l3	1
l2, l4	3
l3, l1	0
l3, l2	0
l3, l3	0
l3, l4	0
l4, l1	0
l4, l2	0
l4, l3	0
l4, l4	0

Table2.14: The set of frequent 2-itemsets with minimum support

Item sets	Support count
l1, l2	3
l2, l4	3

Table2.15: The set of frequent 3-itemsets

Item sets	Support count
l1, l2, l4	2

Table2.16: The set of frequent 3-itemsets with minimum support

Item sets	Support count
l1, l2, l4	2

Table2.17: The set of frequent 4-itemsets

Item sets	Support count
Null	-

This example also shows that, support counts of items are tested at each iteration and the item sets with enough support value passes to the next iteration. When no new item set with enough support is produced, the algorithm stops.

2.1.3.2.3 Clustering

Clustering algorithms aim to group similar web pages. Several distance measures are defined but the most common one is page view count in logs. All sessions are described as vectors. Calculating the distance measures and finding similarity between web pages are complicated in a large web site with lots of sessions and heterogeneous web pages [34]. On the other hand, clustering provides similar user behaviors by grouping them.

2.1.3.3 Pattern Analysis

WUM aims to find intended results. So, analyzing the WUM results is the last step. “Intended results” is a concept based on the used algorithm and web site’s needs. Web page analysts can investigate the average session length, the average two web pages distance on any session or the most frequent patterns including a web page. According to his/her needs, he/she may use these results in a WUM application.

2.1.4 WUM Application Areas

Web usage mining provides many advantages to both users and web page owners. It offers many capabilities in a range area. It can provide personalized marketing to e-commerce sites and their trade volume becomes higher as a result. On the other hand, government agents could also use this technology to fight against terrorism.
A popular area of WUM has been recommender systems recently [24] [28]. Recommender systems are used by many web sites today like amazon.com or imdb.com. It basically recommends user an item, movie or just a link that may be interested by a user. Amazon.com recommends many products to users based on their previous products. On the other hand, imdb.com recommends new films to users based on other films previously watched by this user.
Site improvements could be done according to frequently used patterns thanks to WUM. A web site could be reorganized after these patterns are investigated and this site could be done more user-friendly after this process completed.
Increasing a web site's performance is another important application area of web mining. According to frequently used web pages and frequent user behaviors, the next page which will be requested by user could be detected and load to cache of the server [4]. This will result in quicker response and a web site with better performance.

2.2 Previous Approaches for Session Reconstruction

This part explains previous approaches for the session reconstruction phase. Reconstructing user sessions is an important part of web mining process to discover frequent patterns. This basically consists of two parts, the first one is traditional heuristics and the second one is link based approaches. Traditional heuristics consist of time oriented heuristics and navigation oriented heuristic. On the other hand, Link based heuristics consist of Smart Session Construction Algorithm (SSRA) [8] and Complete Session Construction Algorithm (CSRA) [6]. SSRA misses some of the user sessions and CSRA produces many false sessions in the session reconstruction phase. In order to obtain more accurate sessions, a new algorithm is proposed. New session reconstruction algorithm Link Based Limited Session Reconstruction Algorithm (LSRA) is also a link based heuristic and will be explained in the next chapter.
Web logs consist of users actions on the web. While users are navigating from one page to another, they have more than one option. The first and the most common one is using hyperlinks. Users visit a page and then navigate to another one by using a hyperlink that current page includes. Another way to navigate another page is writing a web site's URL to address bar. These two pages are on the same sessions and they are related to each other. There are two strategies used to detect sessions: The first one is proactive strategies and the next one is reactive strategies. Proactive strategies aim to detect these sessions online while user is browsing on the web page [22, 30], on the other hand, reactive strategies detects these sessions from server logs [31, 15, 16]. Detecting these pages from server log is a difficult

process. Because http protocol is connectionless and stateless, this similarity does not be seen by logs easily. In order to detect that two pages are on the same session, these pages should also belong to the same user. Users mean IP address on the logs and each different IP address represents different users according to reactive strategies. However, one difficulty to separate them is that proxy servers give the same IP numbers to different users and they are seen as the same users on web logs. Another problem is that some of the web page requests by users are not seen on the web logs. This is because of proxy or browser caching mechanisms and if a user requests the same page second time, this page could be get from cache and for that reason server cannot write this request to the log file. Proactive strategies could partially solve these problems on the client sides by using java applets. They get user requests on the client machine but if the user disables this applet, then this approach does not work. On the other hand, link based heuristics also aim to catch these pages by producing possible paths of users by using web topology graph. However, link based strategies are only suitable for static web pages similar to other reactive strategies, they do not predict sessions of dynamic web pages from server logs if a dynamic web page is not written to a log file by the server.

2.2.1 Time Oriented Heuristics

Time oriented heuristics compare page view times while constructing sessions [31, 15]. After separating web entries according to users in the web logs, it cares about their visit time to construct sessions. There are two types of this heuristic. The first one compares visiting time of the first and the last page in a session. If this page is over the determined time bound, then the next page starts new session. The most commonly used upper bound is 30 minutes [12]. If access times of web pages are $[AT_1, AT_2, AT_3 \ldots AT_n]$ and α is the upper bound in a session, then it must be $AT_n - AT_1 < \alpha$.
Here is an example for Time oriented heuristic 1:

Table2.18: Web access times for time oriented heuristic 1

Web Pages	**Access Times**
P1	0m
P2	3m
P3	19m
P4	25m
P5	33m
P6	34m

Upper bound: 30m
According to this information, the time difference between page 4 and page 1 is 25m. P5 could not be inserted to this session because time difference between this page and page 1 is 33m which is over 30m. Thus, our first session is [P1, P2, P3, P4] and our second session is

[P5, P6].
The next time oriented heuristic compares two consecutive pages. According to time oriented heuristic 2, time difference between any consecutive pages must be lower than upper bound. Most commonly accepted upper bound is 10 minutes [12]. If access times of web pages are $[AT_1, AT_2, AT_3 \ldots AT_n]$ and α is the upper bound in a session, then it must be $AT_k - At_{k-1} < \alpha$ where 2<=k<=n.
Here is the example of Time oriented heuristic 2:

Table2.19: Web access times for time oriented heuristic 2

Web Pages	**Access Times**
P1	0m
P2	3m
P3	19m
P4	25m
P5	33m
P6	34m

Upper bound: 10m
According to this information, time difference between page 1 and page 2 is 3m. However, time difference between page 2 and page 3 is 16m which is greater than 10m. The time difference between the rest of the pages are smaller than 10m. Thus, our first session is [P1, P2] and our second session is [P3, P4, P5, P6].

2.2.2 Navigation Oriented Heuristic

Navigation oriented heuristic aims to find sessions according to hyperlinks between web pages. In order to find hyperlinks between them, we need to construct graph of these web pages. It is called web topology graph [5, 19, 26]. Table shows an example of topology graph of web pages: [P1, P2, P3, P4, P5].

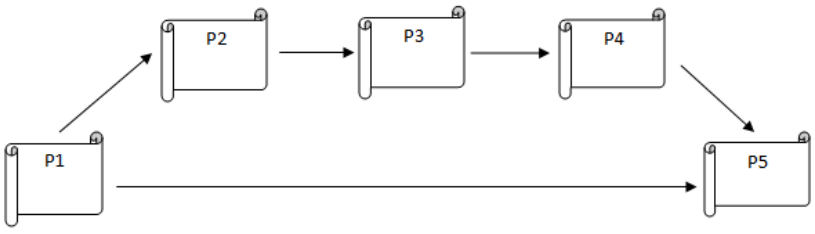

Figure 2.3: Example Web Topology Graph

In a navigation oriented session, each web page except the first one can be reached from one

of the previous pages via hyperlink. This previous page does not have to be the nearest previous page but it has to have smaller access time.
While constructing navigation oriented session, it has two possibilities. Assume that $[P_1, P_2, \ldots, P_m, P_{m+1}, \ldots, P_n]$ are our web pages and P_{n+1} is the next session page.

- If there is a hyperlink from P_n to P_{n+1}, then the next page is added to the end of the session. Then, our session becomes $[P_1, P_2, \ldots, P_m, P_{m+1}, \ldots, P_n, P_{n+1}]$

- If the last page does not have a hyperlink to the new page, then it is tested that the previous page P_{n-1} and the new page P_{n+1} whether there is a hyperlink between them. Pages are tested to the backward until a hyperlink is founded between them. When a page having a hyperlink to the new page is founded, then all pages we tested are added to the end of page before adding new page. For example, if P_m has a hyperlink to P_{n+1}, then our session becomes $[P_1, P_2, \ldots, P_m, P_{m+1}, \ldots, P_n, P_{n-1}, P_{n-2}, \ldots, P_m, P_{n+1}]$.

The major drawback of navigation oriented heuristic is adding these backward movements to the session. These pages simulate backward movements from browser but make the session very long. Another disadvantage of them is that they make hard to interpret pattern.
Here is the example of navigation oriented heuristic:

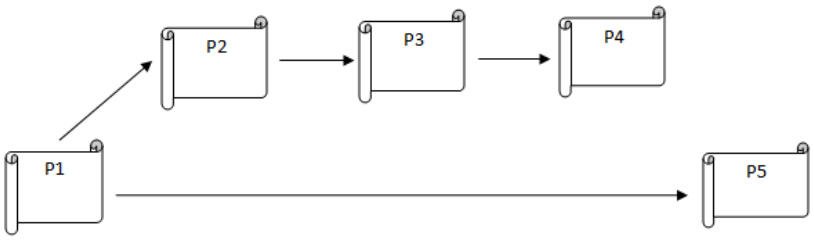

Figure 2.4: Example Web Topology Graph for navigation oriented heuristic

Table2.20: Web access times for navigation oriented heuristic

Web Pages	**Access Times**
P1	0m
P2	3m
P3	19m
P4	25m
P5	33m
P6	34m

Table2.21: Navigation oriented heuristic example

Session	Added Page
[]	P1
[P1]	P2
[P1,P2]	P3
[P1,P2,P3]	P4
[P1,P2,P3,P4]	P5
[P1,P2,P3,P4,P3,P2,P1,P5]	-

2.2.3 Link Based Heuristics

Because of the drawbacks of navigation and time oriented heuristics, topology based algorithms are developed by [7]. The first one is Smart-SRA (Smart Session Reconstruction Algorithm) which is better than traditional heuristics. These algorithms are compared by using simulated and real data. Simulated data is basically produced by using agents in order to compare algorithms. For real data, www.ceng.metu.edu.tr web logs are used. These comparison methods are also used for our new approach and will be explained in the next chapter. Another topology based approach is Complete – SRA (Complete Session Reconstruction Algorithm) which uses a different session reconstruction algorithm compared to Smart-SRA. Complete-SRA results are not as good as Smart-SRA for simulated data. In this thesis, this algorithm is also compared by using www.ceng.metu.edu.tr web logs to compare with traditional algorithms and Smart-SRA. Our new algorithms and these comparison methods will be explained in the next chapter in a detailed manner.

2.2.3.1 Smart – SRA

Smart-SRA [7] uses web topology graph to get rid of backward browser movements that are the result of navigation oriented heuristic. This approach consists of two phases. In the first phase, short candidate sessions are produced by using time oriented heuristics rules. The second phase is a little bit more complicated. Second part results in the maximal sub-sessions by using time and referrer rules. More clearly, a maximal sub-session of the result of the second part $[P_1, \ldots, P_i, P_{i+1}, \ldots, P_n]$ has link and timestamp ordering rule. Link rule means that any page except the first one has a hyperlink from one of the previous pages. Timestamp ordering rule means that any page except the last one has an access time lower than the next page. This rule guarantees to prevent session from backward movements. According to timestamp rule, the access time of the next session page cannot be greater than predefined limit. The predefined limit is 10 minutes accorting to SSRA.

Table 2.23 shows an example of SSRA. In the first iteration, P1 is inserted to the temp page set from candidate session and new session is constructed. Then, P2 is taken in the second iteration and added to first session because there is a hyperlink from P1 and P2. After that,

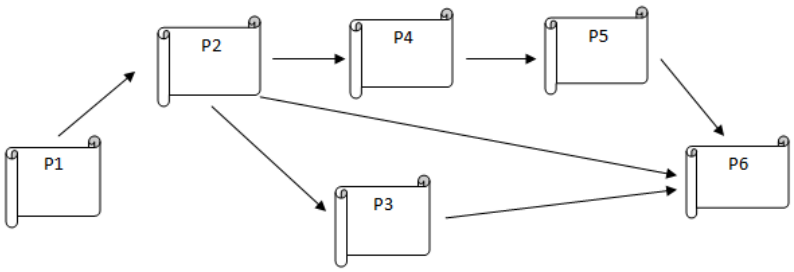

Figure 2.5: Example Web Topology Graph for link based heuristics

Table2.22: Web access times for link based heuristics

Web Pages	**Access Times**
P1	0m
P2	3m
P3	5m
P4	6m
P5	8m
P6	11m

Table2.23: Example of Smart-SRA algorithm

Iteration	**1**	**2**	**3**	**4**	**5**
Candidate Session	[P1,P2,P3, P4,P5,P6]	[P2,P3,P4, P5,P6]	[P3,P4,P5, P6]	[P5,P6]	[P6]
New Session Set (before)		[P1]	[P1,P2]	[P1,P2,P3], [P1,P2,P4]	[P1,P2,P3], [P1,P2,P4,P5]
Temp Page Set	P1	P2	P3,P4	P5	P6
Temp Session Set	[P1]	[P1,P2]	[P1,P2,P3], [P1,P2,P4]	[P1,P2,P4,P5]	[P1,P2,P3,P6], [P1,P2,P4,P5,P6]
New Session Set	[P1]	[P1,P2]	[P1,P2,P3], [P1,P2,P4]	[P1,P2,P3], [P1,P2,P4,P5]	[P1,P2,P3,P6], [P1,P2,P4,P5,P6]
Explanation	P1: Start page	P1 has link to P2	P2 has link to both P3 and P4	P4 has link to P5, but P3 does not have.	Both P3 and P5 has link to P6

P3 and P4 are inserted to temp page set together from candidate session because there is no hyperlinks between them. P2 has a hyperlink to both page and we have 2 sessions at the end of the third iteration. Finally, P5 and P6 are added to the suitable sessions in the next iterations. According to the example, the discovered maximal sessions by Smart-SRA are [P1, P2, P3,

P6] and [P1, P2, P4, P5, P6].
Note that in the first phase, while candidate sessions are being generated, if the access time difference is over 10 minutes between two pages, a new candidate session is generated. This new session is also generated if the total duration of the first session is over 30 minutes. The conclusion of the second part is discovering the user's navigation over hyperlinks in a web topology. For this reason, each previous page has a link to the next page in the maximal sessions and no backward movements are inserted similar to navigation oriented heuristic. Second part of the algorithm has also time rule, which means that time difference between any two consecutive pages has to be smaller than 10 minutes according to SSRA.
While comparing heuristics, we will be interested in two concepts in the next chapter. The first one is recall. Recall for an algorithm is the ratio of number of correct maximal sessions over the total number of correct maximal sessions. The second one is called precision of an algorithm. Precision is the ratio of number of correct maximal sessions over the number of all maximal sessions generated by an algorithm. We will also compare heuristics according to the geometric mean of precision and recall which gives accuracy value. Using geometric mean while computing accuracy means that, for an algorithm, it is important that an algorithm should not only find correct sessions but also avoid generating false sessions.
Note that in the second phase of Smart-SRA, while constructing temp page set, this algorithm only produces pages which do not have a hyperlink to another page in the temp page set. This prevents the algorithm from producing too many sessions and decreases the total number of false sessions generated by this heuristic. However, this approach also prevents session from finding some of the correct sessions. The next approach Complete - SRA is designed to find all possible maximal sessions in a web graph.

2.2.3.2 Complete – SRA

Complete Session Reconstruction Algorithm (CSRA)[6] also consists of 2 phases. Similar to S-SRA, in the first phase candidate sessions are produced by using page stay time and session duration time. The aim of the second phase is constructing maximal patterns according to C-SRA. The details of second phase are the following:

- Each page in the candidate session is processed from start to end in order to determine whether that page is added to an existing sequence or that page starts a new sequence. While extending the existing sequence, there must be link between the last page of the sequence and the new page. Time rule is also checked at this point. Thus, time difference between these pages can not be greater than 10 minutes. While processing these pages, if no new sequence can be produced from a sequence by adding new pages, this session is added to final sequences set. After each page is processed in the candidate session, maximal sessions in the temporary sequences are also added to final session set. Then the final sequence for a candidate session is added to global maximal sequences set.

Below is the example for C-SRA according to table. Navigation sequences are shown by using <*sequence, degree (the number of new sequences that can be constructed from that sequence by adding new pages to its last page), maximality flag*>

Table2.24: Example of Complete-SRA algorithm-1

Iteration	**1**	**2**	**3**
Page	P1	P2	P3
Temp Sequences		<[P1],1,T>	<[P1,P2],3,T>
Extended Set		<[P1],0,F>	<[P1,P2],2,F>
New Sequence	<[P1],1,T>	<[P1,P2],3,T>	<[P1,P2,P3],1,T>
Final Set			
Description	P1 is initial page	P2 is the next page P1 has a link to it	P3 extends the sequence [P1,P2]

Table2.25: Example of Complete-SRA algorithm-2

Iteration	**4**	**5**
Page	P4	P5
Temp Sequences	<[P1,P2],2,F> <[P1,P2,P3],1,T>	<[P1,P2],1,F> <[P1,P2,P3],1,T> <[P1,P2,P4],1,T>
Extended Set	<[P1,P2],1,F>	<[P1,P2,P4],0,F>
New Sequence	<[P1,P2,P4],1,T>	<[P1,P2,P4,P5],1,T>
Final Set		
Description	P4 again extends [P1,P2] and constructs new sequence	P5 extends [P1,P2,P4] and this makes the out degree of [P1,P2,P4] 0. So we delete [P1,P2,P4] from temp sequences.

According to the example, the discovered maximal sessions by Complete-SRA are [P1, P2, P6], [P1, P2, P3, P6] and [P1, P2, P4, P5, P6]. Notice that all pages obey the time rules. The first page is P1 and new sequence is created with P1. Its maximality value is true because it is a new page and it has 1 outer degree because there is a 1 hyperlink from P1 to other pages according to web topology graph. The second page is P2 and the sequence [P1] is extended. The new sequence is produced as [P1, P2]. Its outer degree is 3 because P2's outer degree is 3 and its maximality value is true. The sequence [P1] is deleted from temp sequences because it is extended and its outer degree is decreased to 0 now. It cannot be extended anymore. Then, the new page P3 is processed. The sequence [P1, P2] is extended to [P1, P2, P3]. After that, the new page P4 is processed and this page extends [P1, P2] to [P1, P2, P4] according to topology and this decreases the outer degree of [P1, P2] to 1. Thus [P1, P2] still remains in the Temp sequences. The new page P5 extends the sequence [P1, P2, P4]. Now [P1, P2, P4]'s outer degree is 0 which means that it is deleted from Temp sequences. The new sequence [P1, P2, P4, P5] is added to Temp sequences. After that, the new processed page is P6. P6 extends

Table2.26: Example of Complete-SRA algorithm-3

Iteration	**6**	**7**	**8**
Page	P6	P6	P6
Temp Sequences	<[P1,P2],1,F> <[P1,P2,P3],1,T> <[P1,P2,P4,P5],1,T>	<[P1,P2,P3],1,T> <[P1,P2,P4,P5],1,T>	<[P1,P2,P4,P5],1,T>
Extended Set	<[P1,P2],0,F>	<[P1,P2,P3],0,F>	<[P1,P2,P4,P5],0,F>
New Sequence	<[P1,P2,P6],0,T>	<[P1,P2,P3,P6],0,T>	<[P1,P2,P4,P5,P6], 0,T>
Final Set	<[P1,P2,P6],0,T>	<[P1,P2,P6],0,T> <[P1,P2,P3,P6],0,T>	<[P1,P2,P6],0,T> <[P1,P2,P3,P6],0,T> <[P1,P2,P4,P5,P6], 0,T>
Description	P6 extends [P1,P2]. This makes the out degree of [P1,P2] 0. So we delete [P1,P2] from temp sequences.	P6 extends [P1,P2,P3]. This makes the out degree of [P1,P2,P3] 0. So we delete [P1,P2,P3] from temp sequences.	P6 extends [P1,P2,P4,P5]. This makes the out degree of [P1,P2,P4,P5] 0. So we delete [P1,P2,P4,P5] from temp sequences.

the sequence [P1, P2] and makes the new sequence [P1, P2, P6]. Since the new sequence's outer degree is 0, it is directly inserted to the final sequences instead of the temp sequences because it cannot be extended now. Finally P6 also extends sequences [P1, P2, P3] and [P1, P2, P4, P5] and new sequences [P1, P2, P3, P6] and [P1, P2, P4, P5, P6] are added to final sequences.

The basic property of Complete-SRA is that it searches all of the possible sequences until no more out degree exists from its last page. Therefore, this algorithm finds any possible sequences in a candidate session which obeys navigation and time rules. The advantage of this behavior is that the number of correct sessions of a user is increased compared to S-SRA. However, the number of false session count increases dramatically compared to S-SRA because it finds all of the possible sequences. Clearly it has an important negative effect on the algorithm's accuracy value.

CHAPTER 3

LINK BASED LIMITED SESSION MODEL

3.1 Session Reconstruction with Link Based Limited Model

As stated in the previous chapter, both of the link based session reconstruction algorithms have some characteristic properties to find correct user sessions and patterns. We are using accuracies of different heuristics while comparing them. Accuracy is geometric mean of recall and precision as stated in the previous chapter. This means that both precision and recall must be high in order to increase the accuracy value of a heuristic. Recall value is directly related to the number of correct sessions of an algorithm. On the other hand, precision is directly related to the number of session count that is found by a heuristic. Thus, not only finding correct sessions but also avoiding producing false sessions is important to increase accuracy value. Although the second heuristic Complete-SRA is more successful to capture correct sessions than Smart-SRA, Its precision value is not good due to the number of false sessions it founds. Thus, we do not find all the possible sequences in the second phase of link based model and we need to put some limitations while discovering sessions from candidate session according to web topology graph. In this thesis, the new heuristic puts limitations while reconstructing sessions and tries to capture correct patterns. In the results section, this heuristic is compared to traditional time and navigation heuristics, Smart-SRA and Complete-SRA.
The first phase of the algortihm is the same with the other link based heuristics SSRA and CSRA. After finding candidate sessions according to time and navigation rules in the first phase, algorithm finds the sessions in the second phase. Second phase is shown at Algorithm 1. In the pseudocode, limit function is used to compare parameters with threshold values. These parameters are changed according to applied algorithm. Limit function returns true if needed parameters obeys the limitation rule; otherwise, it returns false.

Algorithm 1 Second Phase of Algorithm

$Globalvariables : MaximalSequenceSet := \{\}$
//for each candidate session
for each $CandidateSession \in CandidateSessionSet$ **do**
  $TemporarySequenceSet := \{\}$
  $FinalSequenceSet := \{\}$
  for each $P_i \in CandidateSession$ // P_i is i-th web page. **do**

$initializeNewPageFlag := FALSE$

8: **for each** $Sequence_j \in TemporarySequenceSet$ //$Sequence_j$ is the j-th temporary sequence **do**

$NewSequence.degree := P_i.outdegree$

10: $NewSequence.maximalityFlag := TRUE$

$NewSequence := [P_i]$

12: **if** $NewSequence.degree = 0$ **then**

$FinalSequenceSet := FinalSequenceSet \cup \{NewSequence\}$

14: **else**

$TemporarySequenceSet := TemporarySequenceSet \cup \{NewSequence\}$

16: **end if**

end for

18: **if** $initializeNewPageFlag = FALSE$ **then**

$linkStatus := Link[LastElement(Sequence_j), P_i]$

20: $nodeCount := PageOrderOfSequence(P_i, CandidateSession) - PageOrderOfSequence(LastElement(Sequence_j), CandidateSession)$

$totalNodeCount := nodeCount + Sequence_j.totalNodeCount - 1$

22: $timeDifference := AccessTime(P_i) - AccessTime(LastElement(Sequence_j))$

if $nodeCount > 1$ **then**

24: $totalTimeDifference := timeDifference + Sequence_j.totalTimeDifference$

else

26: $totalTimeDifference := Sequence_j.totalTimeDifference$

end if

28: limitResult := Limit()

if linkStatus=true **and** limitResult=true **then**

30: $initializeNewPageFlag := TRUE$

$Sequence_j.degree := Sequence_j.degree - 1$

32: $Sequence_j.maximalityFlag := FALSE$

$NewSequence.degree := newSequence(Sequence_j \bullet P_i, totalNodeCount, maximalityFlagTrue, P_i.outdegree, totalTimeDifference)$ //Append new page to the end of Sequence

34: **if** $NewSequence.degree = 0$ **then**

$FinalSequenceSet := FinalSequenceSet \cup \{NewSequence\}$

36: **else**

$TemporarySequenceSet := TemporarySequenceSet \cup \{NewSequence\}$

38: **end if**

if $Sequence_j.degree = 0$ **then**

40: $TemporarySequenceSet := TemporarySequenceSet - \{Sequence_j\}$

end if

42: **end if**

end if

44: **end for**

for each $Sequence_j \in TemporarySequenceSet$ **do**

46: **if** $Sequence_j.maximalityFlag = TRUE$ **then**

$FinalSequenceSet := FinalSequenceSet \cup Sequence_j$

48: **end if**

end for
$MaximalSequenceSet := MaximalSequenceSet \cup FinalSequenceSet$
end for

3.1.1 Time Limited Link Based Model

According to Time Limited Link Based Algorithm, Limit function compares the last page of a sequence access time and the i-th page of the candidate session access time. If the time difference between them is over the determined limit, then this page does not append to that sequence. CSRA and SSRA algorithms also controls time differences in the second part similar to the first part of these algorithms. However, time limit is decreased to lower than 10 minutes in the time limit function contrary to first part of these algorithms. Thus, this prevents many pages from appending the existing sequence and causes starting new sequences.

Function Limit()

if $timeDifference < timeThreshold$ **then**
return true
else
return false
end if
EndFunction

To illustrate, assume that the first phase of the algorithm results in the candidate session [P1, P2, P3, P4, P5, P6] and the web topology graph is as shown in Figure 2.3 again. The access times of the web pages are given below for the time limited link based algorithm. For the node limited link based algorithm, again Tabel 3.1 is used in which the access time difference of two successive pages is not over the threshold limit 4 minutes.

Table3.1: Web access times for time limited link based algorithm

Web Pages	Access Times
P1	0m
P2	3m
P3	5m
P4	6m
P5	8m
P6	11m

Table3.2: Example of Time Limited link based algorithm-1

Iteration	**1**	**2**	**3**
Page	P1	P2	P3
Temp Sequences		<[P1],1,T>	<[P1,P2],3,T>
Extended Set		<[P1],0,F>	<[P1,P2],2,F>
Limit Function		timeDifference(P1,P2) = 3m < 4m, true	timeDifference(P2,P3) = 2m < 4m, true
New Sequence	<[P1],1,T>	<[P1,P2],3,T>	<[P1,P2,P3],1,T>
Final Set			
Description	P1 is initial page	P2 is the next page P1 has a link to it. Time difference of two page is 3m which is lower than 10m. Thus, LimitFunction returns true.	P3 extends the sequence [P1,P2]

Table3.3: Example of Time Limited link based algorithm-2

Iteration	**4**	**5**	**6**
Page	P4	P5	P6
Temp Sequences	<[P1,P2],2,F> <[P1,P2,P3],1,T>	<[P1,P2],1,F> <[P1,P2,P3],1,T> <[P1,P2,P4],1,T>	<[P1,P2],1,F> <[P1,P2,P3],1,T> <[P1,P2,P4,P5],1,T>
Extended Set	<[P1,P2],1,F>	<[P1,P2,P4],0,F>	<[P1,P2],1,F>
Limit Function	timeDifference(P2, P4) = 3m < 4m, true	timeDifference(P4, P5) = 2m < 4m, true	TimeDifference(P2, P6) = 8m < 4m, false
New Sequence	<[P1,P2,P4],1,T>	<[P1,P2,P4,P5],1,T>	
Final Set			
Description	P4 again extends [P1,P2] and constructs new sequence	P5 extends [P1,P2,P4] and this makes the out degree of [P1,P2,P4] 0. So we delete [P1,P2,P4] from temp sequences.	The limit function returns false because the time difference between two page is over 4m. So **there is no new sequence**.

Table3.4: Example of Time Limited link based algorithm-3

Iteration	**7**	**8**	**9**
Page	P6	P6	
Temp Sequences	<[P1,P2],1,F> <[P1,P2,P3],1,T> <[P1,P2,P4,P5],1,T>	<[P1,P2],1,F> <[P1,P2,P3],1,T> <[P1,P2,P4,P5],1,T>	<[P1,P2],1,F> <[P1,P2,P3],1,T>
Extended Set	<[P1,P2,P3],1,F>	<[P1,P2,P4,P5],0,F>	
Limit Function	timeDifference(P3,P6) = 6m < 4m, false	timeDifference(P5,P6) = 3m < 4m, true	
New Sequence		<[P1,P2,P4,P5,P6], 0,T>	
Final Set		<[P1,P2,P4,P5,P6], 0,T>	<[P1,P2,P3],1,T> <[P1,P2,P4,P5,P6], 0,T>
Description	The limit function returns false because the time difference between two page is over 4m. So **there is no new sequence**.	P6 extends [P1,P2,P4,P5] because the time limit function returns true. This makes the out degree of [P1,P2,P4,P5] 0. So we delete [P1,P2,P4,P5] from temp sequences.	Finally, the sequences with true maximality flag in the temp sequences added to final set. Here [P1,P2,P3] is added.

As a result, the sequences [P1, P2, P3], [P1,P2,P4,P5,P6] are the final sequences that algorithm finds.

3.1.2 Node Limited Link Based Model

Node Limited Link Based Algorithm states that if the count of pages between the new page and the last page of the sequence is over the threshold, then new page does not append to that sequence.

Function Limit()

if $nodeCount < nodeThreshold$ **then**
  return true
else
  return false
end if
EndFunction

The basic property is the threshold value that affects the behavior of this function. Threshold

value determines page count between last element of the sequence and the new page. If the threshold value is small, then this means that navigation can only be made between pages which are close to each other in a session. In other words, we eliminate the sequences that include pages distant to each other.

There are many different threshold approaches tried in this thesis work. Mainly 3 different approaches are used to determine this threshold value. The first one is setting the threshold value to a constant value. This approach is not a good solution because candidate session size is not constant. The constant threshold value which may be OK for short candidate session may be inappropriate for long candidate session. The second one is dependent to candidate session page count. This approach set the threshold value according to the candidate session that is produces by phase 1 of the algorithm. Some of the used threshold values are candidate session count / 2, candidate session count / 3 and candidate session count / 4. The last approach depends on the extended sequence size. According to this approach, threshold value updates dynamically and as the extended sequence is getting longer in the iterations of the algorithm, threshold value is becoming higher and higher. Some of the used threshold values of this approach are extended sequence count, extended sequence count / 2 and extended sequence count / 3.

In this example the Limit function uses the candidate session count / 2 value as the threshold value. Because our candidate session is [P1, P2, P3, P4, P5, P6], the count of pages is 6. Candidate session count over 2 results in 3. So, our threshold value is 3 for that candidate session.

Table3.5: Example of Node Limited link based algorithm-1

Iteration	**1**	**2**	**3**
Page	P1	P2	P3
Temp Sequences		<[P1],1,T>	<[P1,P2],3,T>
Extended Set		<[P1],0,F>	<[P1,P2],2,F>
Limit Function		nodeCount(P1,P2) = 1 < 3, true	nodeCount(P2,P3) = 1 < 3, true
New Sequence	<[P1],1,T>	<[P1,P2],3,T>	<[P1,P2,P3],1,T>
Final Set			
Description	P1 is initial page	P2 is the next page P1 has a link to it. Page difference of two page is 1 which is lower than 3. Thus, the limit function returns true.	P3 extends the sequence [P1, P2]

Table3.6: Example of Node Limited link based algorithm-2

Iteration	**4**	**5**	**6**
Page	P4	P5	P6
Temp Sequences	<[P1,P2],2,F> <[P1,P2,P3],1,T>	<[P1,P2],1,F> <[P1,P2,P3],1,T> <[P1,P2,P4],1,T>	<[P1,P2],1,F> <[P1,P2,P3],1,T> <[P1,P2,P4,P5],1,T>
Extended Set	<[P1,P2],1,F>	<[P1,P2,P4],0,F>	<[P1, P2],1,F>
Limit Function	nodeCount(P2,P4) = 2 < 3, true	nodeCount(P4,P5) = 1 < 3, true	nodeCount(P2,P6) = 4 < 3, false
New Sequence	<[P1,P2,P4],1,T>	<[P1,P2,P4,P5],1,T>	
Final Set			
Description	P4 again extends [P1, P2] and constructs new sequence	P5 extends [P1, P2, P4] and this makes the out degree of [P1, P2, P4] 0. So we delete [P1, P2, P4] from temp sequences.	The limit function returns false because there are 4 pages between them which is over 3. So **there is no new sequence**.

Table3.7: Example of Node Limited link based algorithm-3

Iteration	**7**	**8**	**9**
Page	P6	P6	
Temp Sequences	<[P1,P2],1,F> <[P1,P2,P3],1,T> <[P1,P2,P4,P5],1,T>	<[P1,P2],1,F> <[P1,P2,P3],1,T> <[P1,P2,P4,P5],1,T>	<[P1,P2],1,F> <[P1,P2,P3],1,T>
Extended Set	<[P1,P2,P3],1,F>	<[P1,P2,P4,P5],0,F>	
Limit Function	nodeCount(P3,P6) = 3 < 3, false	nodeCount(P5,P6) = 1 < 3, true	
New Sequence		<[P1,P2,P4,P5,P6], 0,T>	
Final Set		<[P1,P2,P4,P5,P6], 0,T>	<[P1,P2,P3],1,T> <[P1,P2,P4,P5,P6], 0,T>
Description	The limit function returns false because there are 3 pages between them which is equal to 3. So **there is no new sequence**.	P6 extends [P1,P2,P4,P5] because the page limit function returns true. This makes the out degree of [P1,P2,P4,P5] 0. So we delete [P1,P2,P4,P5] from temp sequences.	Finally, the sequences with true maximality flag in the temp sequences added to final set. Here [P1,P2,P3] is added.

Notice that we can also set our threshold value as a constant value or a variable depending on the extended sequence count. Firstly, if we set the value of threshold as constantly 3, this algorithm results in the same sequences according to example. However 3 is small for a candidate session that includes many pages. Another approach is setting the threshold value according to extended sequence. Notice that the threshold value is changed at each iteration according to this approach. For example, if extended sequence count is used as threshold value, the threshold value is becoming 2 at iteration 3 for extended sequence [P1, P2] or becoming 5 at iteration 8 for extended sequence [P1,P2,P4,P5].
According to example, assume that the first phase of the algorithm results in the candidate session [P1, P2, P3, P4, P5, P6] and the web topology graph is Figure 2.5 again. The access times of the web pages are given above for the time limited link based algorithm. For the node limited link based algorithm, again Table 3.1 is used.

3.1.3 Link Based Model With Hybrid Limitation

Link Based Model With Hybrid Limitation uses both time and node limitation while constructing sequences. Moreover, this limitation function states that total node count and total time difference must be smaller than predefined threshold in order to return true. Total node count is the sum of node counts between consecutive nodes in the sequence. Similarly, total time difference is the sum of all time differences between the access times of pages which are consecutive in the sequence but there is at least one page between them in the candidate session. Remember that node count and time difference are calculated before the new page is appended to the session between the last page of the sequence and the new appending page.

Function Limit()

---

if $timeDifference < timeThreshold$ **and** $nodeCount < nodeThreshold$ **and** $totalTimeDifference < totalTimeThreshold$ **and** $totalNodeCount < totalNodeThreshold$ **then**
  return true
else
  return false
end if
EndFunction

An example of link based algorithm with hybrid limitation is shown at the table. For this example threshold values are the followings:
nodeCount threshold: Length of extended sequence + 1. This value is changed at each iteration and shown in the table.
totalNodeCount threshold: Length of candidate session / 2. Our candidate session is [P1, P2, P3, P4, P5, P6], so it's length / 2 is equal to 3.
timeDifference threshold: 4m
totalTimeDifference threshold: 10m

Table3.8: Example of link based algorithm with hybrid limitation-1

Iteration	**1**	**2**	**3**
Page	P1	P2	P3
Temp Sequences		<[P1],1,T,0,0m>	<[P1,P2],3,T,0,0m>
Extended Set		<[P1],0,F,0,0m>	<[P1,P2],2,F,0,0m>
nodeCount threshold (Extended Set length + 1)	1	2	3

Limit Function		(nodeCount(P1,P2) = 1) < 2, true;
		totalNodeCount = 0 < 3, true;
		timeDifference = 3m < 4m, true;
		totalTimeDifference = 0m < 10m, true;
			(nodeCount(P2,P3) = 1) < 3, true;
			totalNodeCount = 0 < 3, true;
			timeDifference = 2m < 4m, true;
			totalTimeDifference = 0m < 10m, true;

New Sequence	<[P1],1,T,0,0m>	<[P1,P2],3,T,0,0m>	<[P1,P2,P3],1,T,0,0m>
Final Set			
Description	P1 is initial page	P2 is the next page P1 has a link to it. Node count between two page is 0 which is lower than 3. Total node count is 0(< 3). Total Time difference between two page is 3m(< 4m). Total time difference is 0m(< 10m). Thus, limit function returns true.	Limit function returns true, P3 extends the sequence [P1, P2]

Table3.9: Example of link based algorithm with hybrid limitation-2

Iteration	**4**	**5**	**6**
Page	P4	P5	P6
Temp Sequences	<[P1,P2],2,F,0,0m> <[P1,P2,P3],1,T,0,0m>	<[P1,P2],1,F,0,0m> <[P1,P2,P3],1,T,0,0m> <[P1,P2,P4],1,T,1,3m>	<[P1,P2],1,F,0,0m> <[P1,P2,P3],1,T,0,0m> <[P1,P2,P4,P5],1,T,1,3m>
Extended Set	<[P1,P2],1,F,0,0m>	<[P1,P2,P4],0,F,1,3m>	<[P1,P2],1,F,0,0m>
nodeCount threshold (Extended Set length + 1)	3	4	3

Limit Function	(nodeCount(P2,P4) = 2) < 3, true;
	totalNodeCount = 2 < 3, true;
	timeDifference = 3m < 4m, true;
	totalTimeDifference = 3m < 10m, true
		(nodeCount(P4,P5) = 1) < 4, true;
		totalNodeCount = 2 < 3, true;
		timeDifference = 2m < 4m, true;
		totalTimeDifference = 3m < 10m, true
			(nodeCount(P2,P6) = 4) < 3, false;
			totalNodeCount = 4 < 3, false;
			timeDifference = 8m < 4m, false;
			totalTimeDifference = 8m < 10m, true

New Sequence	<[P1,P2,P4],1,T,1,3m>	<[P1,P2,P4,P5],1,T,1,3m>	
Final Set			
Description	P4 again extends [P1, P2] and constructs new sequence	P5 extends [P1, P2, P4] and this makes the out degree of [P1, P2, P4] 0. So we delete [P1, P2, P4] from temp sequences.	The limit function returns false because some of the limit parameters return false. So **there is no new sequence**.

Table3.10: Example of link based algorithm with hybrid limitation-3

Iteration	**7**	**8**	**9**
Page	P6	P6	
Temp Sequences	<[P1,P2],1,F,0,0m> <[P1,P2,P3],1,T,0,0m> <[P1,P2,P4,P5],1,T,1, 3m>	<[P1,P2],1,F,0,0m> <[P1,P2,P3],1,T,0,0m> <[P1,P2,P4,P5],1,T,1, 3m>	<[P1,P2],1,F,0,0m> <[P1,P2,P3],1,T,0, 0m>
Extended Set	<[P1,P2,P3],1,F,0,0m>	<[P1,P2,P4,P5],0,F,1, 3m>	
nodeCount threshold (Extended Set length + 1)	4	5	
Limit Function	(nodeCount(P3,P6) = 3) < 4, true; totalNodeCount = 3 < 3, false; timeDifference = 6m < 4m, false; totalTimeDifference = 6m < 10m, true	(nodeCount(P5,P6) = 1) < 5, true; totalNodeCount = 2 < 3, true; timeDifference = 3m < 4m, true; totalTimeDifference = 3m < 10m, true	
New Sequence		<[P1,P2,P4,P5,P6],0 ,T,1,3m>	
Final Set		<[P1,P2,P4,P5,P6],0, T,1,3m>	<[P1,P2,P3],1,T,0, 0m> <[P1,P2,P4,P5,P6], 0,T,1,3m>
Description	The limit function returns false because total node count is 3 which is not smaller than limit. So **there is no new sequence**.	P6 extends [P1,P2,P4,P5] because the page limit function returns true. This makes the out degree of [P1,P2,P4,P5] 0. So we delete [P1,P2,P4,P5] from temp sequences.	Finally, the sequences with true maximality flag in the temp sequences added to final set. Here [P1,P2,P3] is added.

In the table, navigation sequences are shown by using <*sequence, degree (the number of new sequences that can be constructed from that sequence by adding new pages to its last page), maximality flag, total node count, total time difference*>

When we compare the link based algorithms according to the same web topology graph and candidate session, Table 3.11 gives the comparison of the results. Notice that the major problem of Complete –SRA is finding all possible sequences and this dramatically increases the false session count although this makes its true session count high. Our aim is decreasing the

ratio of false session count over total count of founded sessions. The algorithm's approach is that it eliminates the sessions that includes pages which are not close to each other. This approach states that the possibility of user's navigation to the new page from one of the recent pages is higher than older pages in the candidate session. Thus, we eliminate the sequences that include pages that a user navigates from older page to the newer page. We claim that users are mostly navigating the new page from the recent pages. Thus, by using this approach, we can decrease the false session counts and this situation results in higher precision. Thus, we expect higher accuracy value.

Table3.11: Comparison of link based algorithms

	S-SRA	**C-SRA**	**L-SRA**
Final Sessions	[P1,P2,P3,P6], [P1,P2,P4,P5,P6]	[P1,P2,P6], [P1,P2,P3,P6], [P1,P2,P4,P5,P6]	[P1,P2,P3], [P1,P2,P4,P5,P6]

3.2 DISCOVERING PATTERNS

After finding sessions, the second part of the preprocessing phase is completed. Preprocessing is the last part of the data cleaning phase. The next phase is pattern discovery. This phase aims to find frequent user access patterns from the sessions that are found in the previous chapter. We are interested in sequential pattern mining methods because time of the pages is important for us. As we mention in the sequential pattern mining part, we have some alternatives including GSP [32], SPADE[37]. We used AprioriAll[3] algorithm sequentially. AprioriAll is the most appropriate algorithm for link based session reconstruction heuristics [8].
There are several reasons that make Sequential Appriori appropriate for our restrictions. One reason is the exact matching of sequences. According to our domain, a pattern is only supported by a session if and only if a session's subsequence exactly matches that pattern. For example, when we test whether a pattern [P1, P3] is supported by a sequence [P1, P2, P3], we can see that P1 comes before P3 according to both sequence. However, the pattern [P1, P3] is not supported by the session [P1, P2, P3] and P2 disrupt the matching. On the other hand, another pattern [P2, P3] is supported by that session [P1, P2, P3] according to Sequential Appriori Algorithm. Another reason is that, our topological hyperlink constraint removes most of the possible candidate sessions while constructing them and this makes the performance of Sequential Appriori Algorithm very good.
Before explaining the algorithm, an important concept, support function, is explained. Support function determines whether a pattern is supported from the given sessions. Support of the session is calculated by the count of sessions that supports the pattern divided by total count of sessions. Sequential AprioriAll Algorithm starts with finding supported patterns with length 1. Algorithm test all of the pages whether they have enough support value. Then, these supported patterns constitute our supported patterns with length 1. After that, until no more supported pattern is produced, we do the following iteration. Each supported pattern

with length k is tested for every page in the domain whether it has a hyperlink to it. If it has link to page P, then the new candidate pattern with k+1 is produced. After that, we test whether this candidate pattern has enough support. If it has enough support, then its maximality flag is set to true. On the other hand, the following pattern's maximality flags are set to false:

- The pattern with length k and the pattern with length 1 that constitute new length k+1 pattern.
- The pattern produced by removing the first page of the new k+1 supported pattern if it is in the supported patterns set.

When no more supported pattern is produced, iterations stop and algorithm passes to final step. In this step, the algorithm eliminates non-maximal patterns and keeps only maximal patterns in the final supported patterns list. Notice that we first control that whether there is a hyperlink from the pattern with length k to a page P before we test the support value at any iteration. This makes the algorithm performance better and protects the algorithm from unnecessary support value calculations at each step.

To illustrate, the following small example shows the frequent patterns of SSRA, CSRA and LSRA by using Table 3.11. Remember that our candidate session is [P1, P2, P3, P4, P5, P6]. Table 3.11 states that sessions are [P1, P2, P6], [P1, P2, P3, P6] and [P1, P2, P4, P5, P6] according to CSRA, so there are 3 sessions in this domain. Assume that support value is 0.6. Therefore, if a pattern is contained by at least 2 of these sessions, it has enough support; otherwise, support function returns false. Firstly, pseudo code finds supported patterns with length 1. The set of web pages in the domain consists of {P1, P2, P3, P4, P5, P6}. Thus, each web page in the domain is tested whether it has enough support. P1, P2 and P6 is contained by all of the sessions. This means that these pages' support value is 1. P3, P4 and P5 are contained by only one of these sessions and their support values are 0.33. They are eliminated because their support values are smaller than 0.6. After that, supported patterns with length 2 are found in the second iteration. Remember that the set of supported patterns with length 1 is P1, P2, P6. Each page in this set is tested with all of the pages in the domain whether there is a link between these pages. For example, P1 and P2 has link, thus P2 is appended to P1 and [P1, P2] is a new candidate pattern now. After that, this new candidate pattern is tested by support function. [P1, P2] is also contained by all of the sessions and support function returns true. Therefore, this makes maximality flag of [P1, P2] true and [P1] false. Now, the first page of [P1, P2] is dropped and the new candidate pattern without first page is [P2]. The maximality flag of this pattern is set to false because the set of supported patterns with length 1 includes this pattern. Finally, [P1,P2] is added to the set of supperted patterns with length 2. In the iteration 2, all of the combinations between the supported patterns with length 1 and all pages in the domain are tested. At the end of the iteration 2, the set of supported patterns with length 2 only contains [P1, P2] in this example. The next step is testing candidate patterns with length 3. However, none of the candidate pattern has enough support value. At the end of

the algorithm, only patterns with true maximality flag are insreted to final maximal patterns. Thus, [P1] and [P2] are eliminated for that reason. As a result, final maximal frequent patterns of CSRA are [P1,P2] and [P6] according to support value 0.6.
These algortihm steps are repeated for the results of LSRA and SSRA. Results are shown in the Table3.12. Result of CSRA and SSRA contain [P1,P2] and [P6]. On the other hand, [P6] does not have enough support value according to LSRA. Thus, LSRA only contains [P1,P2] as a maximal frequent pattern.

Table3.12: Comparison of frequent patterns

	S-SRA	**C-SRA**	**L-SRA**
Final Sessions	[P1,P2,P3,P6], [P1,P2,P4,P5,P6]	[P1,P2,P6], [P1,P2,P3,P6], [P1,P2,P4,P5,P6]	[P1,P2,P3], [P1,P2,P4,P5,P6]
Maximal Frequent Patterns	[P1,P2], [P6]	[P1,P2], [P6]	[P1,P2]

3.3 AGENT SIMULATOR

In the experimental results section, we will compare the algorithms by using both simulated and real data. In order to make comparison between heuristics, we need to have real user navigation paths. Each user requests cannot be totally determined by using server logs as we explained in the first chapter. The fact that some of the user requests are replied by proxy or client caches is the most important reason for this problem. For this reason, we need to simulate web user behaviors. An agent simulator is designed and implemented to compare S-SRA to traditional heuristics by [10]. We have also implemented this by regulating some of the concepts. The agent simulator produces a web topology graph randomly and then creates user agents for navigating on that graph by using power law property [13, 29]. The agent simulator also simulates the local cache replies of a client. Thus, it does not write entry to server log file if that page is in the client history. For example, the real session of a user [P1, P2, P1, P3] can be written to server log file as [P1, P2, P3]. Because the second access of P1 is replied from the client caches. Because agent simulator knows the real navigation paths of simulated users, the heuristic's results can be compared to correct results by using this way.
The agent simulator has some parameters in order to define a user's behavior. While simulating a user, these parameters determine the user's next action.
Session Termination Probability: STP value determines the probability that the user terminates the current session.
Link from Previous pages Probability: LPP value determines that the user navigates to the new page from a previously visited page in that session. This previously visited page has to be one of the pages that this user previously visited in this session except the last visited page. While selecting the previous page, the probability of selection of recent pages is higher than

older pages. This parameter also simulates the backward movements in the browser.
Link from Current page Probability: LCP value determines that the user navigates to the new page from the last visited page. This parameter simulates that the user navigates to the new page from the current page that has a hyperlink to the new page.
New Initial page Probability (NIP): NIP value is the probability of navigating a new page that has no hyperlink from the current page. This behavior simulates to write new URL to the address bar on the browser.
The agent simulator first decides whether it continues to current session with p probability or ends the current session with 1 – p probability according to random surfer model of page ranking algorithm. If it decides to continue the current session with probability p, then it decides whether to select a hyperlink from the current page by using the probability of LCP, or presses backward button from the browser and select a hyperlink from previous pages by using the probability of LPP. Another option for the simulator is that it can end the current session with probability (1 - p). Its first alternative is writing a URL to the address bar and navigate a web page that current page does not have a hyperlink to by using the probability of NIP. Its second alternative is ending the current session by using the probability of STP. It is also important that the time difference cannot be more than 10 minutes if the session continues according to agent simulator. Thus, it gives the new page's access time smaller than the current page's access time plus 10 minutes. The simulator also does not give the access time of a page more than 30 minutes of the access time of the first page in a session according to time oriented heuristic rules. Agent simulator constructs a server web log file at the end of the process by using its sessions.
For example, the following session simulates an example behavior of the agent according to our domain by using Table 2.22 and Figure 2.5. According to table, our session is [P1, P2, P3, P4, P6]. The session starts with P1. Then, if the random function of the simulator selects LCP, user navigates to P2. After that, assume that random function again selects LCP value. This means that the user navigates the new page from the current page that has a hyperlink to the next page. We have three alternatives which are P3, P4 and P6. Assume that the random function selects P3. Now, the random function selects LPP value. This means user navigates the new page from one of the previous pages that has a hyperlink to the new page. Remember that recent pages have higher probability than older pages to be selected. Simulator selects P2. According to this selection, user hit backward button one times on the browser and navigates to P2 and then navigates to P4 because P4 is the only page that has a hyperlink from P2 according to our domain. At this point, our agent simulator selects the NIP probability value and selects P6 that has no hyperlink from P4. Finally, simulation ends with the selection of STP value.

3.4 WEB TRACKER TOOL

In order to use real data for comparing heuristics, we have to handle correct user sessions for a web site. Web tracker tool is designed for this purpose. It consists of two parts. The first part is a JavaScript code working on the client machine. This part collects user's navigations

paths at runtime and writes these paths to the web server logs with special id. The second part of this tool is a web log parser that is designed to parse the server web log file and gets the true sessions of users by collecting and parsing web log entries that are written by web tracker client tool from the web server logs.
The JavaScript function of the tracker tool is needed to be placed in a web page. While a user request that page from the server, this function also works and requests a 1x1 pixel image with user's session parameters. The server logs this request to the web log file with the received parameters. These parameters include user's information. This function needs to be placed at each web page that user's actions are intended to be traced. Here is an example server web log entry of the tracker tool:

```
88.227.142.123 - - [15/Jul/2012:18:48:14 +0300] "GET /grad/px.gif?
_sID=http://www.ceng.metu.edu.tr/grad/curriculum&
_vID=Frl8wlS9SfipiPNtg2VCaA%3D%3D&
_ssID=0mCkjv6nRQiM16OBOAKRog%3D%3D&
_url=http%3A%2F%2Fwww.ceng.metu.edu.tr%2Fgrad%2Fcurriculum&
_tt=Graduate%20Curriculum%20%5BMETU%20Computer%20Engineering%5&
_rf=http%3A%2F%2Fwww.ceng.metu.edu.tr%2Fundergrad%2Fcourses%3
Fcrsprogram%3Dall&_ln=tr&_cd=32&_sr=1366x768&_ic=true&_ij=true&
_fv=11.2%20r202&_cs=utf-8&_tv=&_rnd=70%2FEqIgV HTTP/1.1"
200 4070 "http://www.ceng.metu.edu.tr/grad/curriculum" "Mozilla/4.0 (compatible;
MSIE 7.0; Windows NT 6.1; Trident/5.0; SLCC2; .NET CLR 2.0.50727; .NET CLR
3.5.30729; .NET CLR 3.0.30729; eSobiSubscriber 2.0.4.16; .NET4.0C; .NET4.0E)"
```

Figure 3.1: Web tracker log entry

This is a regular web log entry. The interesting part of the entry is get request section. This section holds the information of the user. The name of the image that is requested from the server is "/grad/px.gif". Important parameters send with that image are shown below:

```
/grad/px.gif?_sID=http://www.ceng.metu.edu.tr/grad/curriculum&
_vID=Frl8wlS9SfipiPNtg2VCaA%3D%3D&
_ssID=0mCkjv6nRQiM16OBOAKRog%3D%3D&
_url=http%3A%2F%2Fwww.ceng.metu.edu.tr%2Fgrad%2Fcurriculum&
_tt=Graduate%20Curriculum%20%5BMETU%20Computer%20Engineering%5&
_rf=http%3A%2F%2Fwww.ceng.metu.edu.tr%2Fundergrad%2Fcourses%3Fcrsprogram
%3Dall&_ln=tr&_cd=32&_sr=1366x768&_ic=true&_ij=true&
_fv=11.2%20r202&_cs=utf-8&_tv=&_rnd=70%2FEqIgV
```

Figure 3.2: Information part of a web tracker log entry

_sID (http://www.ceng.metu.edu.tr/grad/curriculum): The URL of the web page that tracker is placed in.
_vID (Frl8wlS9SfipiPNtg2VCaA%3D%3D): Visitor id of the user. This id consists of random characters and unique for each different users. It is used to recognize the same user.
_ssID (0mCkjv6nRQiM16OBOAKRog%3D%3D): Session id of the user. This part is used to detect the session that this web page belongs to.
_rf (http%3A%2F%2Fwww.ceng.metu.edu.tr %2Fundergrad%2Fcourses%3Fcrsprogram%3 Dall): The referrer page.
_rnd (70%2FEqIgV): Random bytes for that server request. For each request of this image, different random bytes are added to the URL as parameter. Thus, no two requests have the same URL and for that reason the request does not replied from the local cache. Therefore, we guarantee that each request is logged on to the web log file.

Notice that this px.gif image is an extra server request for each web page but its cost is minimal because it is a 1x1 small image.

The tracker tool has a simple logic. While it is assigning session id to user requests it checks whether there exists a cookie for this user. if there is a cookie, it uses current session id of this cookie, if not, it assigns a random session id and writes it to the cookie. In the event that user uses another browser, which implies user has started to a new session, tracker tool gives a new session id for that user since the cookies of the browsers are specific to each other.

The most useful advantage of the tool is catching the pages that are replied from client's cache. When a user requests a page second time with the same URL, it is replied from the client's cache and this means that request could not be logged to the server's web log file. On the other hand, tracker tool attaches random id to the end of the URL while requesting px.gif image, thus any request is replied by server instead of client's cache even if it has the same URL with one of the previous one.

As an example, assume that user requests pages of [P1,P2,P3,P4,P3,P1,P5]. Since the pages of P4, P3 and P1,which are written as bold, are requested second time, they are replied by client cache and this whole sequence is reflected to server logs as [P1,P2,P3,P4,P5].However the tracker tool saves this sequence as [P1,P2,P3,P4,P3,P1, P5].

The second part of the web tracker tool is parser tool. The parser tool takes the web server log files as an argument. The result of the parser is correct sessions file. The parser recognizes the trace tool entries by px.gif image file. After collecting these entries, it takes web page URL, visitor ID, session ID of the page. Then by using visitor and session information, it distributes the pages to different sessions with respect to their access times. While constructing sessions, parser tool also obeys the time rule. That means, the access time difference between two consecutive pages have to be smaller than 10 minutes in a session although both has the same session id and user id. Similarly, the access time difference of the first and the last page of the session cannot be more than 30 minutes according to parser tool. These user sessions which

consist of user's navigation paths are printed to a file in order to be used to make comparison between heuristics.

CHAPTER 4

EXPERIMENTAL RESULTS

In this chapter, we are going to compare heuristics by using both simulated and real data. The compared heuristics are traditional heuristics which are time oriented heuristics and navigation oriented heuristic, link based heuristics which consist of S-SRA, C-SRA and our new algorithm. Algorithms are shown in the Table 4.1. LSRA-1 is Node Limited Session Reconstruction Algorithm. The node count threshold in limit function is set to length of candidate session / 2 for Node Limited SRA. On the other hand, LSRA-2 uses hybrid limitation. Threshold values are the followings:
nodeCount threshold: Length of extended sequence + 1.
totalNodeCount threshold: Length of candidate session / 2.
timeDifference threshold: 4 minutes
totalTimeDifference threshold: 10 minutes

Table4.1: Parts of web request explanations

Abreviations in figures	**Algorithm name**
TO	Time Oriented Approach
NO	Navigation Oriented Approach
SSRA	Smart Session Reconstruction Algorithm
CSRA	Complete Session Reconstruction Algorithm
LSRA-1	Limited Session Reconstruction Algorithm (Node Limited)
LSRA-2	Limited Session Reconstruction Algorithm (Hybrid Limitation)

4.1 Accuracy Metric

The next part is showing the simulated data comparison of the heuristics. The agent simulator worked firstly. The agent simulator parameters LPP, LPC, STP and NIP are changed at each run. 1000 different users are simulated at each run. The average pages stay time is set to 2.2 minutes in the simulator. Then, simulator produces the web topology graph, a server web log file contains the web logs of the simulated users and another file consists of true sessions

of the users. After that, the different algorithms work by using produced web log file. Each different algorithm produces correct sessions file. We compare each file with the true sessions file to find accuracy of each heuristic. Firstly, precision of each algorithm is found. Precision is the ratio of the count of the captured true sessions over the count of the sessions that this heuristic produces. Then recall value is calculated. Recall value is the ratio of the count of true sessions captured by this heuristic over the count of true sessions that our agent simulator produces. We can compute the accuracy value by using geometric means of precision and recall.

$$\text{RECALL} = \frac{|SessionsProducedbyHeuristic \cap TrueSessionsProducedByAgent|}{|SessionsProducedbyHeuristic|} \tag{4.1}$$

$$\text{PRECISION} = \frac{|SessionsProducedbyHeuristic \cap TrueSessionsProducedByAgent|}{|TrueSessionsProducedByAgent|} \tag{4.2}$$

$$\text{ACCURACY} = \sqrt{\text{RECALL} \times \text{PRECISION}} \tag{4.3}$$

Frequent patterns of the heuristics are also compared. Each sessions file produced by a heuristic is processed by Sequential Apriori Algorithm. Sequential Apriori Algorithm is also applied to the correct sessions file which is produced by agent simulator. Then the resulted session files are compared by using their accuracies which is explained above. Two sessions in different sessions' file are compared by exact matching rule. In other words, if the all pages in a session and their orders are the same, then we can say that both of the sessions are equal.
In the comparison on the real data part, results of the heuristics on the real data are compared. As explained in the trace part, web tracker tool is inserted to www.ceng. metu.edu.tr web pages. After information is collected by using web tracker, real user sessions are produced by web tracker parser. Then heuristics are compared by using this web site's web server logs.

4.2 Comparison on Simulated Data

The first result is the comparison of heuristic's precision, recall and accuracy values. The agent simulator LPP / LPC and STP / NIP values are varied from 0.25 to 4. Each combination of these values is given to the simulator at each run and the averages of the results are calculated. At figure 4.1, the results are given. Notice that link based heuristics SSRA and LSRA results have much better accuracy values than traditional time oriented and navigation oriented algorithms. Their accuracy values are nearly 50%. Figure shows that CSRA has the most successful recall value. This means that CSRA captures most of the true sessions of the

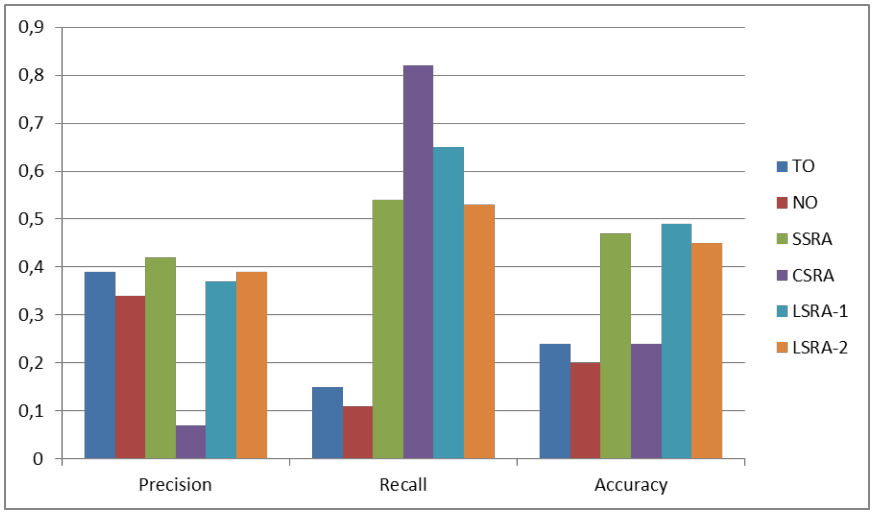

Figure 4.1: Precision, Recall and Accuracy values of Algorithms

simulated users. However, the count of false sessions that are produced by that algorithm is very high. That results in very low precision value. Thus, the accuracy value of that algorithm is not very good. Finally, our new algorithm LSRA finds true sessions less than CSRA because of its restrictions. On the other hand, it produces false sessions much lower than CSRA and this increases the accuracy value of LSRA. The accuracy value of LSRA-1 is also slightly better than SSRA for the simulated data.
The next phase of our experiments is processing heuristic results by Sequential Apriori Algorithm to find frequent user patterns. Sequential Apriori Algorithm process the heuristic's results and the correct sessions result that is produced by Agent Simulator. After this operation, these results are compared. There are 4 different support values used for Sequential Apriori Algorithm while finding frequent patterns varying from 0.001 to 0.00001. The STP/NIP and LPP/LPC values are set to 4 for this operation. The Figure 4.4, 4.2 and 4.3 shows the accuracy, precision and recall values of the heuristics for different support values for Sequential Apriori Algorithm.

The Figure 4.2 shows that as the support values are getting lower, the precision value of LSRA is not decreasing similar to SSRA. On the other hand, CSRA, navigation and time oriented heuristic precision values are becoming lower. Moreover, The Figure 4.3 shows the recall values of the heuristics for different support values. CSRA and LSRA-1 has definitely better recall values than other heuristics since they catch most of the supported patterns.
The next figure shows the accuracy values of heuristics depending on LPP, LPC, STP, and NIP parameters of agent simulator. The support values for Sequential Apriori are set to 6 different values from 0.01 to 0.00001 at each runs. LPP/LPC and STP/NIP values are varied from 0.25 to 4. Each combination of these values is found at each run and finally the average accuracy values are calculated. The figure 4.5 shows this results.

These results also show that the results of link based heuristics are better than navigation ori-

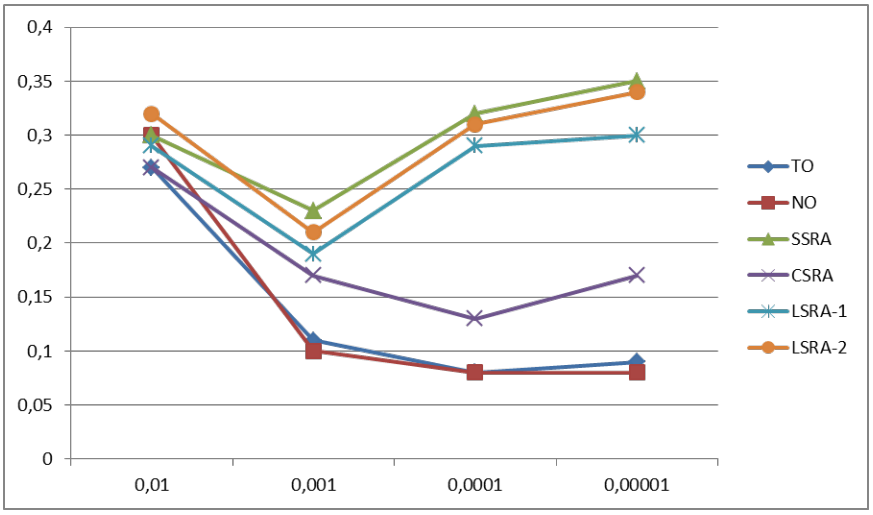

Figure 4.2: Precision values for different support values of Sequential Apriori

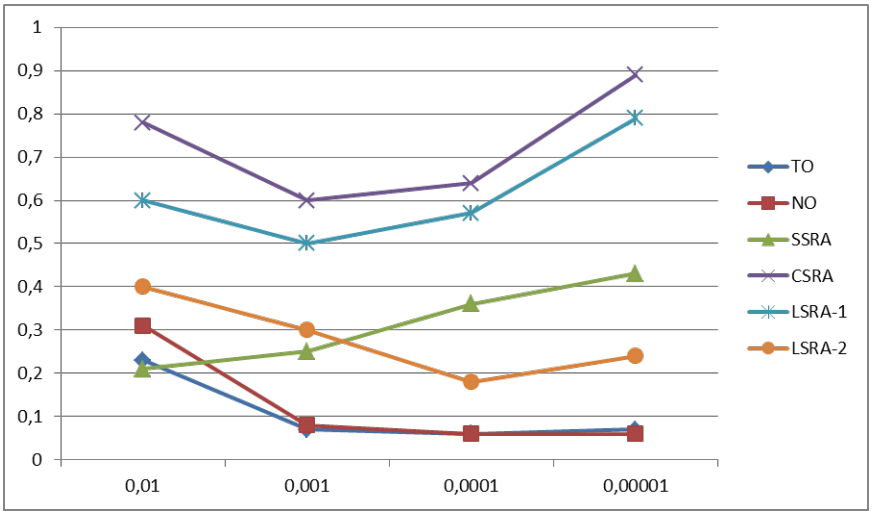

Figure 4.3: Recall values for different support values of Sequential Apriori

ented and time oriented heuristics. Accuracy value of LSRA is slightly better than SSRA. Notice that LPP / LPC value shows the user behavior while navigating between web pages. If LPP / LPC > 1, it means that users are mostly navigate from previous pages and they use backward button. On the other hand, LPP / LPC < 1 means that users mostly navigate from the current page to the new page. STP / NIP values also determine the characteristic of users. The lower STP means the longer user sessions. On the other hand, the higher NIP value means that user mostly navigates from address bar by writing URL address instead of using hyperlinks from the current page. The results of LSRA and CSRA show that both algorithms are stable on different parameter values compared to other heuristics.

Finally, Figure 4.6 shows the average accuracy values of all parameters. This figure also shows that average accuracy result of LSRA-1 is much better than time oriented and naviga-

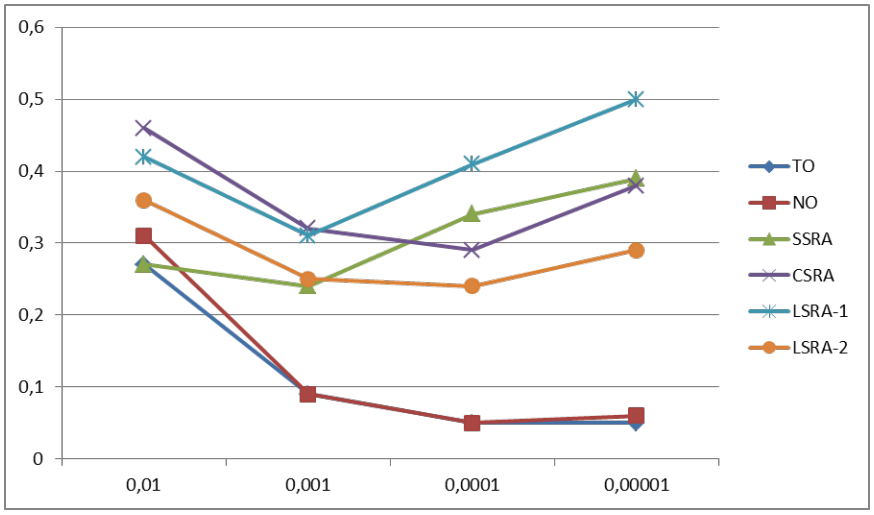

Figure 4.4: Accuracy values for different support values of Sequential Apriori

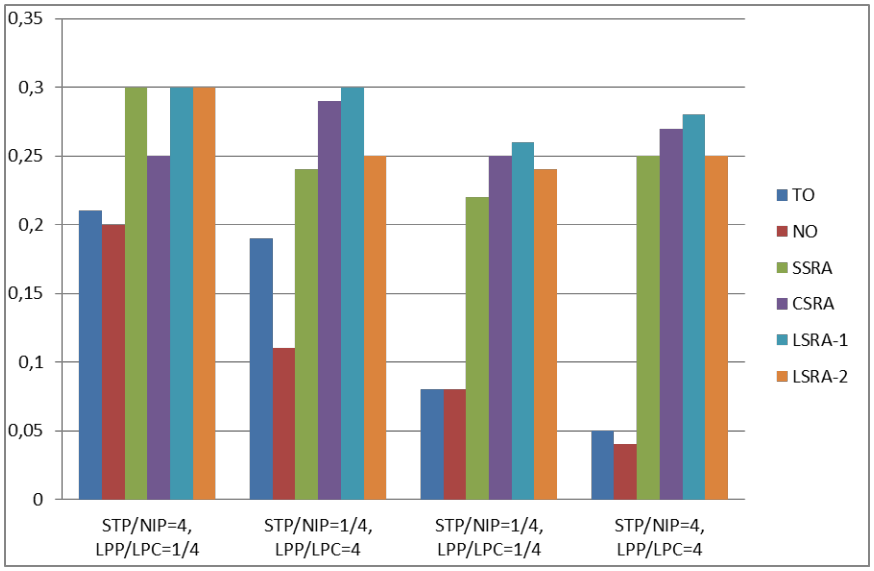

Figure 4.5: Accuracy values for different parameters of agent simulator

tion oriented heuristics and slightly better than SSRA. It is also better than CSRA thanks to better precision value. On the other hand, the average accuracy value of LSRA-2 is slightly worse than SSRA.

4.3 Comparison on Real Data

We have also evaluated the accuracy of the heuristics by using www.ceng.metu.edu.tr web server logs. We tracked the user's actions between July and September 2012. There are 7

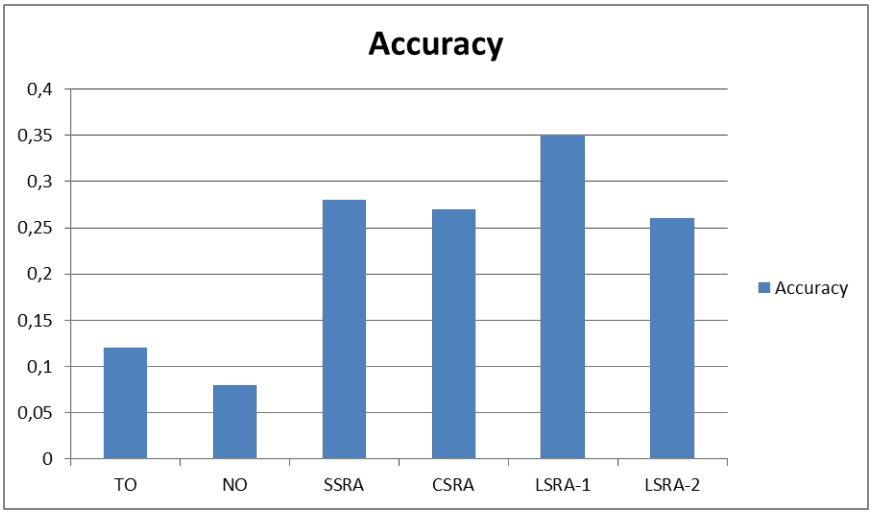

Figure 4.6: Average Accuracy Values

server web log files with 600 MB total size between this period of time. We collect the entries under /undergrad path. The web pages under this path consist of 11 web pages and each page is connected to other pages. So the web topology graph is highly connected. The tracking tool is used which is explained previously. Some of the web page's internal structures are changed and web tracker tool is inserted into html page code. Data of users navigating on the web pages under /undergrad folder of the department are collected. Table 4.2 shows these web pages.

Table4.2: Parts of web request explanations

Page Number	**Page Address**
Page 1	/undergrad
Page 2	/undergrad/index
Page 3	/undergrad/curriculum
Page 4	/undergrad/electives
Page 5	/undergrad/prospective
Page 6	/undergrad/staj
Page 7	/undergrad/courses
Page 8	/undergrad/undefined
Page 9	/undergrad/index#Undergraduate_Program
Page 10	/undergrad/index#Minor_and_Double_Major_Programs
Page 11	/undergrad/curriculum#Double-major_program

After the navigations of visitors are collected, the true sessions are parsed by using web tracker parser tool. This operation gives the true sessions file. Then, the server web log files are collected between these dates. After cleaning of these log files, each heuristics is processed the log files. This process generates session's files each one is produced by different heuristic. After gathering session files, Sequential Apriori Algorithm is applied to these files with differ-

ent support values in order to find frequent patterns. Figures 4.7, 4.8 and 4.9 shows the result of this comparison between heuristics. The first table shows the result of session comparisons of algorithms. This table shows precision, recall and accuracy values of heuristics. The second one is for various support values of frequent patterns. Support values are changed from 0.05 to 0.001. The last table shows the precision, recall and accuracy values of the frequent patterns for support value 0.01.

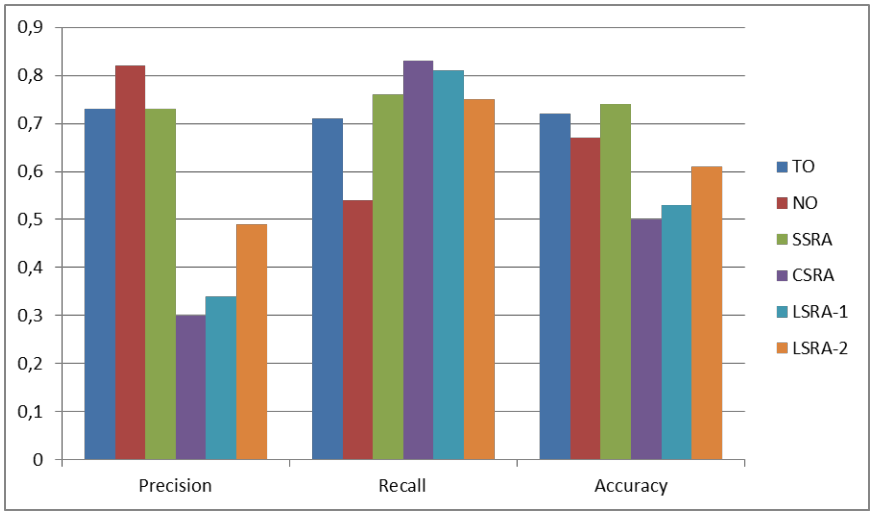

Figure 4.7: Precision, Recall and Accuracy values for heuristics

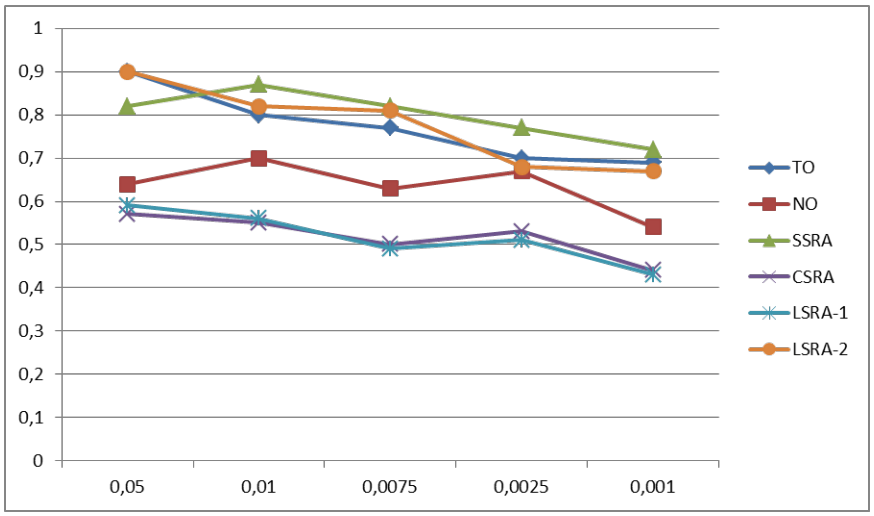

Figure 4.8: Accuracy values for various support values of Frequent Patterns

According to these tables, experiment results are not seemed well but some improvements could be noticed. Results show that SSRA has better accuracy compared to other algorithms. Moreover, SSRA has better frequent pattern results for various support values. The accuracy values and the frequent pattern results of LSRA-2 is better than LSRA-1. Notice that our new algorithm's precision value is better than C-SRA as observed in the previous section. On the

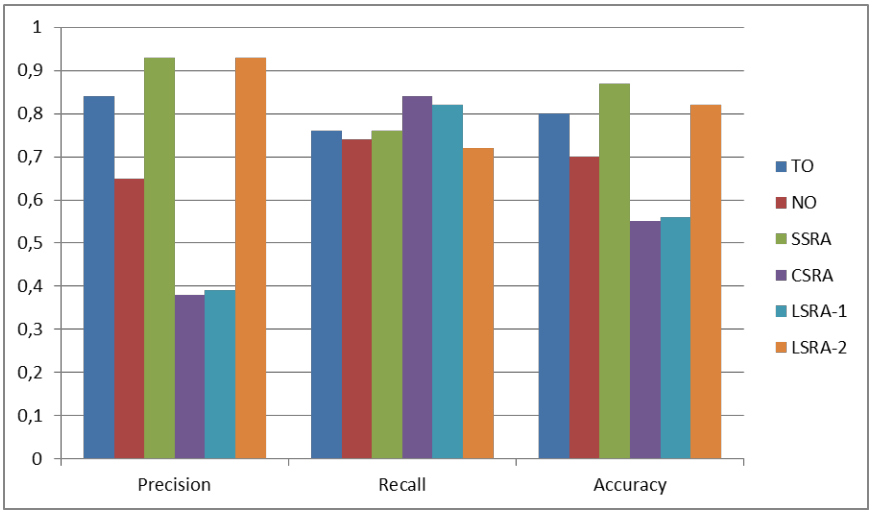

Figure 4.9: Precision, Recall and Accuracy values for support value 0.01 of Frequent Patterns

other hand, the accuracy value of the new algorithm is not better than S-SRA since it misses some of the true frequent patterns due to limitations in the algorithm. The recall value of LSRA-2 is worse than LSRA-1, because it has more limit parameters than LSRA-1. On the other hand, precision value of LSRA-2 is highly better than LSRA-1, because it produces less false sessions compared to LSRA-1.

The reasons for this result are observed as three main issues when the true sessions of users are investigated. The first one is the session definition that web tracker used. In the parsing step of web tracker tool, it distributes pages to different sessions by using time rules. So, if two pages has the same session id, user id and obey the time rules, the tool insert these pages to the same session. This increase the accuracy values of time oriented heuristic positively. Secondly our web topology graph is highly dense. Every page has link to other pages in the domain. Thus, CSRA and LSRA are tries to find all possible paths although LSRA has some limitations. Thirdly, Users' navigation behaviors seem very short. Some sessions also consist of the same repeating pages. The new algorithm and other link based algorithms need long sessions to select correct navigation paths. Time oriented heuristic seems well for short sessions because it does not need long navigation paths for users. For a another web domain where users have long navigation sequences, like shopping web sites or e-commerce sites, results could be better than this for that reason. In spite of these unexpected results, notice that new approach enhances precision and accuracy values of C-SRA and performs slightly better than other approaches except SSRA for various support patterns.

CHAPTER 5

CONCLUSIONS

In this thesis, Link Based Limited Session Reconstruction Algorithm was introduced. Constructing sessions by using web usage logs is a hard problem. Seperating users in this logs can be difficult because different users may have the same IP address due to proxy servers. Another problem is browsers' internal cache mechanisms. Some of the users' page requests are not written into web server logs because browsers reply them from their internal cache. There are traditional session reconstruction algorithms, time oriented and navigation oriented algorithms but they do not solve these problems.

Link based algorithms are developed in order to solve caching problem and construct sessions more accurately. They basically produce possible subsessions by using web topology graph of web sites. Our new algorithm L-SRA is a link based session reconstruction algorithm like the other previously implemented algorithms S-SRA and C-SRA. Both link based heuristics have some problems. The first heuristic S-SRA produces less false user sessions but misses some of the user sessions which lead to low recall value. On the other hand, the second link based heuristic C-SRA captures most of the user sessions but produces false user sessions which lead to low precision value. The new algorithm L-SRA aims to increase the precision value of C-SRA with its new approach and finds web user sessions and frequent user patterns better than not only traditional time and navigation oriented heuristics but also other link based heuristics S-SRA and C-SRA. It puts some limitations while producing subsessions by using a web site's web topology graph. There exists three versions of this algorithm which are time, node and hybrid versions. The new algorithm has better accuracy value by using simulated data compared to other heuristics. On the other hand, it is also tried by using real user data captured from www.ceng.metu.edu.tr, its precision and accuracy values are better than C-SRA and result of various support values for frequent patterns are also better than C-SRA although its accuracy results are not as good as SSRA. It is observed that the result of new heuristic depends on the tracker tool's produced true sessions and user's navigation behavior.

An improved agent simulator is used to produce simulated user sessions. It is used with different parameters to simulate different user behaviors. Many different user behaviors are simulated and the web log files are produced by using agent simulator to compare heuristics on simulated data. After that, frequent patterns of corrects sessions and heuristic results are found. Precision, recall and accuracy values for different heurics are also compared.

Web tracker tool is also implemented to collect www.ceng.metu.edu.tr visitor's sessions by

changing web site's internal structure and inserting web page's html code. By parsing this data, visitor's correct session data is gathered. The most basic advantage of this tool is that it catches the pages replied from client's caches. Heuristics are also compared by using this real data.

Sequential Apriori Algorithm which is an improvement version of Apriori algorithm is used to find user's frequent patterns for each heuristic. It is the most suitable algorithm for link based heuristics because of two reasons. The first one is that we aim to find exact matching while finding supported patterns and this algorithm enables it. The second advantage is algorithm's performance. It only computes linked page's support value and this prevents it from many unnecessary calculations. Comparisons are made by using both user's sessions and user's frequent patterns.

To conclude, our new heuristic is observed as a good algorithm by using simulated data and could be tried as a reactive web usage mining approach to capture user's frequent behavior for web sites which have large domain and users with long navigation path such as e-commerce or shopping web sites.

REFERENCES

[1] R. Agrawal and R. Srikant. Mining sequential patterns: Generalizations and performance improvements. In *Proc. 5th Int. Conf. Extending Database Technology (EDBT'96)*, pages 3–17, Avignon, France.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, Pro-c e e dings of the Twentieth International Conferenc e on V ery Large DataBases, VLDB*, pages 487–499, September 1994.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE*, pages 3–14, 1995.

[4] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira. Characterizing reference locality in the www. In *Proceedings of 1996 International Conference on Parallel and Distributed Information Systems (PDIS '96)*, pages 92–103.

[5] B. Andrei, R. Kumar, and M. Farzin. Graph structure in the web. In *Ninth International World wide web Conference*, Amsterdam, 2000.

[6] M. A. Bayir and I. Hakki Toroslu. Link Based Session Reconstruction: Finding All Maximal Paths. *CoRR abs*, July 2013.

[7] M. A. Bayir, I. H. Toroslu, and A. Cosar. A new approach to reactive web usage data processing. In *WIRI06, the 2nd International Workshop on Challenges in Web Information Retrieval and Integration, ICDE 06's Workshop*, 2006.

[8] M. A. Bayir, I. H. Toroslu, A. Cosar, and G. Fidan. Smart miner: A new framework for mining large scale web usage data. In *WWW*, Madrid, Spain, 2009.

[9] M. A. Bayir, I. H. Toroslu, M. Demirbas, and A. Cosar. Discovering better navigation sequences for the session construction problem. *Data Knowl. Eng.*, 73:58–72, 2012.

[10] M. A. Bayır. A new reactive method for processing web usage data. Master's thesis, METU, 2006.

[11] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifiers (uri): Generic syntax. Network Working Group RFC 2396, `http://www.rfc-editor.org/rfc/rfc2396.txt`, August 1998.

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

[13] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1), 2006.

[14] R. Cooley. *Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data*. PhD thesis, University of Minnesota, 2000.

[15] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1), 1999.

[16] R. Cooley, P. Tan, and J. Srivastava. Discovery of interesting usage patterns fromweb data. In *Advances in Web Usage Analysis and User Profiling*, pages 163–182, LNAI 1836, Springer, Berlin, Germany, 1999.

[17] R. Cooleya, B. Mobasher, and J. Srivastava. Web mining: Information and pattern discovery on the world wide web. In *ICTAI*, pages 558–567, 1997.

[18] R. Cooleya, B. Mobasher, and J. Srivastava. Web mining: Information and pattern discovery on the world wide web. In *Proceedings of the Ninth IEEE International Conference on Tools with Artificial Intelligence(ICTAI'97)*, 1997.

[19] C. Cooper and A. Frieze. A general model of web graphs. In *ESA*, pages 500–511, 2001.

[20] Dinucă, C. Elena, and D. Ciobanu. "web content mining." annals of the university of petrosani. *Economics*, 12(1):85–92, 2012.

[21] N. H. L. Directive. `.http://hoohoo.ncsa.uiuc.edu/docs/setup/httpd/LogOptions.html`, 1995.

[22] Y. Fu and M. Shih. A framework for personal web usage mining. In *International Conference on Internet Computing (IC'2002)*, pages 595–600, Las Vegas, 2002.

[23] He, Daqing, , and A. Göker. Detecting session boundaries from web user logs. In *Proceedings of the BCS-IRSG 22nd annual colloquium on information retrieval research*, pages 57–66, 2000.

[24] M. Jaczynski and B. T. rousse. Www assisted browsing by reusing past navigations of a group of users. In *Proceedings of the Advances in Case-Based Reasoning, Forth Europe an Workshop, EWCBR-98*, pages 160–171, Dublin, Ireland, September 1998.

[25] R. Kosala and H. Blockeel. Web mining research: A survey. *SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG)on Knowledge Discovery & Data Mining, ACM*, 2(1):1–15, 2000.

[26] R. Kumar, R. Prabhagar, and R. Sridhar. The web as a graph. In *19th ACM SIGACT-SIGMOD-AIGART Symp*, 2000.

[27] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tompkins, and E. Upfal. The web as a graph. In *PODS '00: Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–10, 1997.

[28] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on web usage mining. *Communications of the ACM*, 43(8):142–151, 2000.

[29] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. tech. rep. computer systems laboratory. Technical report, Stanford University, Stanford, CA., 1998.

[30] C. Shahabi and F. B. Kashani. Efficient and anonymous web-usage mining for web personalization. *INFORMS Journal on Computing*, 15(2):123–147, 2003.

[31] M. Spiliopoulou and L. Faulstich. Wum: A tool for web utilization analysis. In *Proceedings EDBT workshop WebDB'98, LNCS 1590*, pages 184–203, Springer, Berlin, Germany, 1998.

[32] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *EDBT*, pages 3–17, 1996.

[33] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2):12–23, 2000.

[34] D. Tanasa. *Web Usage Mining: Contributions to Intersites Logs Preprocessing and Sequential Pattern Extraction with Low Support*. PhD thesis, UNIVERSITE DE NICE SOPHIA ANTIPOLIS, 2005.

[35] I. Turner. The one-stop portal. Line56, `http://www.line56.com/articles/default.asp?ArticleID=4075`, October 2002.

[36] W3C. Logging control in w3c httpd. `http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format`, July 1995.

[37] M. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42:31–60, 2001.