5-2008

Mining and tracking evolving web user trends from very large web server logs.

Basheer Hawwash 1984-
University of Louisville

MINING AND TRACKING EVOLVING WEB USER TRENDS FROM VERY LARGE WEB SERVER LOGS

By

Basheer Hawwash

A Thesis
Submitted to the Faculty of the
Graduate School of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

May 2008

Mining and Tracking Evolving Web User Trends From Very Large Web Server Logs

By

Basheer Hawwash

A Thesis Approved on

April 10, 2008

by the following Thesis Committee:

_____

Thesis Director: Dr. Olfa Nasraoui

_____

Dr. Ibrahim Imam

_____

Dr. Suraj Alexander

_____

_____

ABSTRACT

MINING AND TRACKING EVOLVING WEB USER TRENDS FROM VERY LARGE WEB SERVER LOGS

By

Basheer Hawwash

May 10, 2008

Online organizations are always in search for innovative marketing strategies to better satisfy their current website users and lure new ones. Thus, recently, many organizations have started to retain all transactions taking place on their website, and tried to utilize this information to better understand and satisfy their users. However, due to the huge amount of transaction data, traditional methods are neither possible nor cost-effective. Hence, the use of effective and automated methods to handle these transactions became imperative.

Web Usage Mining is the process of applying data mining techniques on web log data (transactions) to extract the most interesting usage patterns. The usage patterns are stored as profiles (a set of URLs) that can be used in higher-level applications, e.g. a recommendation system, to meet the company's business goals. A lot of research has been conducted on Web Usage Mining, however, little has been done to handle

the dynamic nature of web content, the spontaneous changing behavior of users, and the need for scalability in the face of large amounts of data.

This thesis proposes a framework that helps capture the changing nature of user behavior on a website. The framework is designed to be applied periodically on incoming web transactions, with new usage data that is similar to older profiles used to update these old profiles, and distinct transactions subjected to a new pattern discovery process. The result of this framework is a set of evolving profiles that represent the usage behavior at any given period of time. These profiles can later be used in higher-level applications, for instance to predict the evolving user's interest as part of an intelligent web personalization framework.

TABLE OF CONTENTS

ABSTRACT iii

1 INTRODUCTION 1

1.1 Motivation 1

1.2 Objectives 2

1.3 Thesis Contribution 4

1.4 Summary 4

2 LITERATURE REVIEW 6

2.1 Overview of Web Usage Mining 6

2.2 Sources of Web Usage Data 8

2.3 Pattern discovery process 10

2.3.1 Preprocessing usage data 10

2.3.2 Pattern Discovery 13

2.3.3 Patterns Analysis 16

2.4 HUNC 17

2.4.1 Preprocessing the web log file to extract user sessions 18

2.4.2 Assessing Web User Session Similarity 19

2.4.3 The HUNC Algorithm 21

2.5 Evolving Profiles 22

2.6 Scalability 25

2.7 Summary 26

3 METHODOLOGY **27**

3.1 Overview of the Proposed Methodology 28

3.2 Preprocessing the logs 31

3.3 Updating profiles 32

3.4 Discovering distinct profiles 38

3.5 Post-processing the distinct profiles 39

3.6 Combining profiles 39

3.7 Interpreting the profiles 40

3.8 Profile Maintenance 41

3.9 Summary 45

4 EXPERIMENTS **46**

4.1 Evaluation Metrics 46

4.1.1 Profile Variance 47

4.1.2 Profile Cardinality 48

4.1.3 The matching vs. distinct sessions 49

4.1.4 Number of profiles 50

4.1.5 Profile Pair-wise Similarity 51

4.1.6 Profile Density 52

4.2 Parameter configuration 53

4.2.1 Method of discovery 53

4.2.2 Method of matching profiles 54

4.2.3 Similarity threshold 54

4.2.4 URL weight threshold 55

4.3 Experimental Configuration 56

4.4 Experiment 1: University of Missouri 57

4.4.1 The Dataset 57

4.4.2 Profile Variances 60

4.4.3 Profile Evolution 64
4.4.4 Matching vs. Distinct Sessions 67
4.4.5 Profile Cardinality 73
4.4.6 Profile Count 76
4.4.7 Profile Pair-wise Similarity 77
4.4.8 Profile Density 78
4.4.9 Evolution vs. Traditional (Full run) 81
4.5 Experiment 2: University of Louisville Library 85
4.5.1 The Dataset 85
4.5.2 Profile Variances 86
4.5.3 Profile Evolution 96
4.5.4 Matching vs. Distinct sessions 97
4.5.5 Profile Cardinality 101
4.5.6 Profiles Count 104
4.5.7 Profile Pair-wise Similarity 105
4.5.8 Profile Density 107
4.5.9 Evolution vs. Traditional (Full run) 107
4.6 Experimental Design 113
4.7 Summary 115

5 CONCLUSION **117**

REFERENCES **120**

APPENDIX A **125**

CURRICULUM VITAE **144**

LIST OF TABLES

2.1 Log File Sample 18

4.1 Evaluation Metrics Summary (t = time period index) 47

4.2 Experimental Configuration 56

4.3 HUNC Parameters 57

4.4 Missouri: Final Profiles for RobWT (0.6) and URL TH(0.1) 59

4.5 U of L Library: Final Profiles for RobWT (0.6) and URL TH(0.1) . . 87

4.6 Experimental Design Factors 114

4.7 Experimental Design Responses 114

4.8 Experimental Design Data Table 114

LIST OF FIGURES

3.1 Proposed Pattern Discovery Process Flowchart 30

3.2 Sample Profile 39

4.1 Cosine Similarity vs. Profile Variance 48

4.2 Missouri: Access Requests 58

4.3 Missouri: Evolution of Profile Variances (Binary Cosine Similarity) . 61

4.4 Missouri: Evolution of Profile Variances (Robust Weight Similarity) . 63

4.5 Missouri: Profiles Sigma Aggregates 64

4.6 Missouri: Profile Counts (Binary Cosine Similarity) 66

4.7 Missouri: Profile Evolution (Robust Weight Similarity) 68

4.8 Missouri: Matching Sessions Percentage 70

4.9 Missouri: Missing vs. Distinct Sessions (Binary Cosine Similarity) . . 71

4.10 Missouri: Missing vs. Distinct Sessions (Robust Weight Similarity) . 72

4.11 Missouri: Cardinality of Max and Min Sigma 74

4.12 Missouri: Evolution of Profile Cardinality Percentages (CosSim) . . . 75

4.13 Missouri: Evolution of Profile Cardinality Percentages (RobWT) . . 76

4.14 Missouri: Total Profiles Count 77

4.15 Missouri: Profile Pair-wise Similarity Aggregates 78

4.16 Missouri: Evolution of Profile Densities (CosSim) 80

4.17 Missouri: Evolution of Profile Densities (RobWT) 81

4.18 Missouri: Traditional Profile Variances 82

4.19 Missouri: Traditional Profile Density 83

4.20 Missouri: Evolution vs. Traditional Variance Aggregates 84
4.21 U of L Library: Access Requests 86
4.22 U of L Library: Evolution of Profile Variances (CosSim . URL TH(0.04)) 89
4.23 U of L Library: Evolution of Profile Variances (CosSim. URL TH(0.1)) 90
4.24 U of L Library: Evolution of Profile Variances (RobWT . URL TH(0.04)) 92
4.25 U of L Library: Evolution of Profile Variances (RobWT . URL TH(0.1)) 93
4.26 U of L Library: Profiles Sigma Aggregates 95
4.27 U of L Library: Profile Counts (Binary Cosine Similarity) 96
4.28 U of L Library: Profile Counts (Robust Weight Similarity) 97
4.29 U of L Library: Matching vs. Distinct Sessions (Binary Cosine Similarity) 99
4.30 U of L Library: Matching vs. Distinct Sessions (Robust Weight Similarity) 100
4.31 U of L Library: Matching Sessions Percentage 101
4.32 U of L Library: Cardinality of Max and Min Sigma 102
4.33 U of L Library: Evolution of Profile Cardinality Percentages (CosSim) 103
4.34 U of L Library: Evolution of Profile Cardinality Percentages (RobWT) 104
4.35 U of L Library: Total Profiles Count 105
4.36 U of L Library: Profile Pair-wise Similarity Aggregates 106
4.37 U of L Library: Evolution of Profile Densities (CosSim) 108
4.38 U of L Library: Evolution of Profile Densities (RobWT) 109
4.39 U of L Library: Traditional Profile Variances 110
4.40 U of L Library: Traditional Profile Density 111
4.41 U of L Library: Evolution vs Traditional Variance Aggregates 112

CHAPTER 1

INTRODUCTION

1.1 Motivation

The World Wide Web is the largest and most accessible source of information for both users and businesses. Organizations create their own websites to offer their services to the customers (individuals or other companies), and customers flock to these websites to buy new books, rent movies, read news, or perform any kind of business transaction. As more businesses moved online, the competition between businesses to keep their old customers and lure new customers has increased, since a competitor's website is just one click away.

All these reasons have motivated online companies to analyze the usage activity data that results from online transactions in order to better understand and satisfy their website users on an individual or group basis, for instance to support Customer Relationship Management (CRM). However, millions of clicks and online transactions take place every day, and the data that results from these transactions is becoming so huge that trying to use conventional analysis methods is neither possible nor cost-effective. As a result, it has become imperative to use automated and effective methods to turn this raw data into knowledge that can help online organizations to better understand their users.

Web Usage Mining is the process of applying data mining techniques on web log data (transactions) to extract the most interesting users' online activity patterns and extract from them user profiles (e.g. a set of URLs) that can be used in higher-level applications such as recommendation or personalization services, and hence help increase the online users' satisfaction and maintain their loyalty.

There has been a considerable amount of research in web usage mining [1, 3, 4, 6, 14, 15, 18, 19]. However, there has been very few detailed studies in how to deal with the challenging characteristics of today's websites, especially in answering scalability concerns (enormous streams of raw data), dynamic content and the changing behavior of users.

The content of most websites changes on a regular basis that ranges from monthly changes to hourly-changes such as the case of most news websites. The dynamic nature of websites' content along with the social, economical, cultural, and other changes are the driving force for the changing behavior of users over the Web. Since the changing usage behavior is hard to predict, discovering usage patterns should be done dynamically and on a regular basis.

1.2 Objectives

This thesis proposes a new framework that can handle the changing content and behavior of the web users by updating usage patterns' profiles over time based on the new web log data stream. In this framework, web usage mining is performed on a regular basis, such as weekly or daily, or depending on the size of the logs, in case the logs cannot be loaded in memory in their entirely. At each run, the recent user activities are compared against the existing profiles, then the profiles are updated according to a similarity function. The completely new and different usage trends are then used to discover additional new profiles to be added to the older updated profiles.

Following this scalable approach, the usage patterns are expected to always be representative of the current behavioral trends of the users. Moreover, since at any time, only a smaller part of the new log data undergoes the pattern discovery process, this approach is efficient in terms of computational complexity and resource usage. These characteristics make our framework an efficient and robust solution that can be embedded in a higher-level application that utilizes the profiles in intelligent web applications, such as to provide predictions for web page pre-fetching, load balancing, recommendations, or web personalization.

The objectives of this thesis can be summarized as follows:

- Handling the scalability of usage pattern discovery from massive input web usage data.
- Handling the evolution of input usage data resulting from the dynamic nature of the Web content, and the changing behavior of web users.
- Tracking the evolution of the discovered profiles over time by updating an archive of profiles in an on-going basis.
- Generating user-friendly profile representations that can be used in other higher-level applications.
- Evaluating the quality of the proposed profile tracking.

1.3 Thesis Contribution

The major contribution of this thesis is providing an efficient, robust, and scalable approach to discover the most interesting user behavior patterns on a website. This approach is capable of developing and tracking evolving profiles for users' browsing trends on a single website. These profiles are potentially -at any given time- the best representatives of the current users' preferences.

This thesis is also concerned with studying and analyzing the evolution of patterns that are discovered. The approach adopted in this thesis updates the patterns to the smallest details. Hence, observing their evolution will give an accurate picture of how users are really changing their browsing behavior. This analysis may help in pointing out potential business target markets or outdated markets.

This thesis also suggests a maintenance algorithm that will be applied periodically on the discovered patterns to enhance their accuracy and reliability. During maintenance, the patterns are evaluated, thus potentially allowing only interesting or current ones to be retained. Instead of putting every new data record through a complete analysis, a large portion of data (that already match the discovered profiles) will actually be discarded based on appropriate similarity tests against these profiles.

Eventhough, our discussions are within the context of mining web usage data, this approach can be extended to other applications involving mining dynamic data streams, such as mining and summarizing various system and network audit logs, or even temporal text data such as blogs and online news.

1.4 Summary

The motivations behind adopting the proposed framework are the need for analyzing the web users' behavior on websites, and the lack of methods to handle changes efficiently, while adhering to the scalability requirement of this task. We have also presented the main objectives and contributions of this thesis, that is to develop a

robust, scalable, and efficient framework capable of capturing the dynamic usage behavior on the web as a set of evolving profiles, which could be used in higher-level applications to meet a variety of organizational goals.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview of Web Usage Mining

Web Usage Mining is the process of applying data mining techniques to extract useful information usage patterns from the web log data. The analysis of the discovered usage patterns can help online organizations gain many business benefits, including:

- Determining the life time value of customers
- Developing cross-product marketing strategies
- Enhancing promotional campaigns
- Generating automatic recommendations to website visitors
- Web personalization
- Efficient organization of the web sites' content and structure
- Enhancing the server performance by pre-fetching pages that the user is most likely to visit (commonly performed by hosting sites and Internet Service Providers)

The above benefits are not limited to online stores, but they may also impact other types of online "presence", such as e-learning sites and free media websites, including news, publishers and digital libraries.

Web usage mining has gained a lot of interest lately because it focuses on the user's perspective to the Web more than the creators' perspective. As a result, a lot of research has been conducted and many new techniques and tools have been developed [1, 3, 15, 18, 19]. These tools can be divided into two main categories: *Pattern Discovery, and Pattern Analysis*. However, many tools both discover and analyze patterns. This is due to the fact that pattern representation is application-dependent, i.e. there are no standards on how patterns' profiles should look like.

Web Pattern discovery tools are basically what most people refer to as "web usage mining" tools. They use many techniques from different disciplines including Artificial Intelligence, data mining, machine learning, and information theory to infer usage patterns from the web usage data. For example, Nasraoui and Krishnapuram proposed in [14] a new evolutionary clustering approach called Unsupervised Niche Clustering (UNC), which proved robust to noise. However UNC works best with numerical datasets. For many real-world application UNC was not practical. Hence, in [6], a new version of UNC was proposed, named Hierarchical UNC (HUNC), which generates a hierarchy of clusters. These clusters can provide different levels of detail, thus speeding up the mining process considerably. This thesis will provide a modified version of HUNC that can handle the dynamic nature of usage patterns.[5] Proposed a complete framework that automatically classifies visitors to a web site into different classes based on their navigation behavior. Once the visitor is classified, the organization of links will be dynamically changed on the-fly and links will be suggested to the user.

The discovered patterns are not useful unless they are interpreted and understood by the pattern analysis tools. The interpretation of the patterns depend on the tool

and the purpose of that interpretation. for example the WebViz system [18] is used to visualize the web as a connected graph. Web Utilization Miner (WUM) [4] is an example of both a discovery and analysis tool for web usage mining. WUM extracts navigation patterns and represents them by an "aggregate tree" which are browsing trails combined based on their common prefix. Moreover. WUM offers an SQL-based language called MINT that enables the human expert to format the output and choose the interestingness criteria.

2.2 Sources of Web Usage Data

The web usage data can be obtained from different sources: server log files. client side data. and web proxy logs. The most important source is the server log files since they explicitly record the clickstreams for all the users on a particular website and they are much easier to obtain since they are owned by the organization that owns the web servers.

The information available in log files include the client's IP address. time of access. pages visited. previous page visited. and other fields. However. in its raw form. the usage data provided by log files on the web server is not entirely reliable. and does not reflect the actual browsing behavior due to the various levels of caching on the Web. Caching is used by most web browsers to keep a temporary copy on the local machine of most recently visited pages. under the assumption that these pages would most likely be visited again. So when requested. they would be retrieved from the temporary location on the local machine to reduce both the loading time of those pages and the network traffic load. As a result. all the cached pages are not requested from the server. Thus. they are not recorded in the server log file.

Other information that can be found on the server include query data which is the set of keywords that the users search for within that website. as well as cookies

which are small files stored on the local machine and that can be accessed only by the web servers in a special list stored in the cookie. The purpose of cookies is to keep track of users because of the limitation of the stateless connection model of the HTTP protocol. However, cookies need to be enabled from the client side to be useful, and they also raise certain privacy concerns.

On the client side, data can be collected using client-side scripts such as Java Script and Java Applets. However there is a trade-off between the additional computational overhead (such as in Java Applets) and the amount of data collected (such as the sheer amount that Java Script can collect compared to Java Applets). Client-side collection of data solves the problem of caching since the script is embedded in the page itself not on the server side, so every time the page is accessed - even if it is cached - the data will be collected and sent to the server. However, as in cookies, the use of client-side scripts may require user cooperation to be useful, and may also raise some privacy concerns.

Another source of usage data is the web proxy server. A web proxy is basically a server midway between the client and the server, that caches the most recently requested pages in order to reduce the loading time of web pages and reduce the network traffic load. It shares the same concept of caching by web browsers on local machines, however a web proxy is typically shared by multiple users and caches pages from multiple web servers. Hence, a web proxy server can provide accurate and rich information about the common interests of multiple users on multiple websites, which makes it a valuable source for the web usage mining process. On the other hand, local (client-side) caching provides information about only a single user. However, obtaining the data from web proxies is hard and raises more privacy concerns, because web proxies contain the activity of multiple web servers.

The experiments done in this research are based only on web usage data collected from the web server. However, the framework that is presented can easily be extended

to process usage data collected from the client side and proxy servers.

2.3 Pattern discovery process

Analyzing how users access a website can be critical to the organization as discussed earlier, and can be one of the main factors in determining the organization's success or failure in the online world. Acquiring the web usage data is not a hard task since most of the data is on the web servers owned by this organization. However, before this raw data is available for processing by the pattern discovery algorithms, it needs to be prepared/preprocessed for it to be useful.

2.3.1 Preprocessing usage data

Preprocessing the usage data is not a trivial task [1, 2, 19]: it can be arguably the most difficult task in the Web Usage Mining process due to the incompleteness and complexity of data. It is also one of the most important tasks in web usage mining since the discovered patterns are only useful if the usage data gives an accurate picture of the user accesses to the web site. There are two primary tasks in preprocessing: *data cleaning*, and *transaction identification* also known as *sessionization*.

Data Cleaning

During this process, all irrelevant items are eliminated from the server log. An irrelevant item could be an access to a picture, a clip, or anything that is not of any importance to the behavior of the user. Rather, it is embedded within the requested web page. This kind of item could be eliminated by checking the suffix of the URL name and removing all entries ending with GIF, JPEG...etc.

Another kind of irrelevant item comes from requests originating from web agents like search engine bots, crawlers, or spiders. Because these agents access the website periodically to update their search database, each time that they visit the website, a new entry is created in the log. Yet this does not represent an actual user. Hence,

they should be removed. One way to do that is by checking for some known IP addresses or crawler identifiers, and then removing all the corresponding entries from the log.

Transaction Identification

The sequences of page requests must be grouped into logical units to be used in the mining process. Each of these logical units is called a *session* which is the set of pages that are visited by a single user within a predefined period of time. This step needs to be done because sessions describe how users "behave" on the website. Thus, individual URL records are meaningless. This might seems like a trivial process, however, some issues need to be addressed.

One issue is how to determine that the access records really belong to a single user. The problem is that not every single IP address represents exactly one user, and this is due to the fact that ordinary users (normally) do not own a universally unique IP address because of the high cost and low availability of IP addresses. So Internet Service Providers (ISP) own one or more IP addresses and use them to connect their clients to the Internet. ISPs typically have a pool of proxy servers that are used to speed up accessing the Internet by multiple clients, so all these accesses will be seen on the log file as coming from one IP address hence considered one user. Also, some ISPs randomly assign each request from a single user to a different IP address for privacy and optimization purposes (IP address rotation), which means that multiple IP addresses on the log file may actually represent a single user. Moreover, a user accessing the website from different machines may appear as multiple users on the server side.

Some of these cumbersome issues can be solved by using client-side tracking mechanisms such as cookies, or using a combination of IP addresses, machine name, browser agent, and other information. However, client-side tracking needs to be enabled by the client himself, and using a combination of different data (such as IP

address and browsing agent) is not always helpful since even the combination might not be enough to identify individual users (for instance, only a few browsing agents like Internet Explorer and Mozilla Firefox dominate the rest).

Another issue is what the predefined period of time value should be. A 30 minute timeout is often used based on the results of [15], which seems appropriate for websites like online stores, however it might not be the best value for news or publication websites such as *Wikipedia* where one might spend more time reading the content compared to online store.

Cleaning data and transactions identification both modify the content of what the server logs contain to make it available for mining. However determining whether important accesses were are not recorded in the access log is a much harder problem. The main reason for the absence of these records is caching.

Caching is normally used at different levels in the client-server communication module. At the client level, most browsers cache the latest pages visited by the user under the assumption that these pages will be visited again. Caching also happens on the proxy servers of the ISP, where data requested from multiple users is stored temporarily. Caching helps in reducing page loading time on the client side, and reduces the traffic on the server side. So when a user requests a page, the browser search for it in the cache first, if it is not there then it will request it from the server, which also checks the proxy server first for cached data before going to the website web server. As such, all cached pages do not show up on the web server log files which underestimates the actual number of users and requests made.

To overcome this problem client-side scripts such as Java Script and Java Applets can be used, since they are embedded in the page itself. Whenever the page is accessed -even if it is cached- the data will be collected. Another solution is to access the proxy server logs which show exactly which pages were cached and requested again. However, the first approach needs the compliance of the users, and the second

requires access to the proxy servers, generally owned by the Internet Service Providers (ISP), which can raise major privacy issues.

2.3.2 Pattern Discovery

Once sessions have been identified and cleaned, the pattern discovery process can begin. There are several methods that can be used to discover interesting patterns, and these methods are rooted in diverse fields such as Data Mining, Artificial Intelligence, Statistics, Machine Learning, and Pattern Recognition.

Different methods have been developed to discover the patterns, such as *association rule mining, classification, and clustering*. Choosing which method to use should take into consideration any prior knowledge of the Web Data. For example, in association rule mining the notion of a transaction in market-based analysis does not take into consideration the order of items being selected. However, in web usage mining the order of pages requested in the session may be important since it reflects how the user actually reached a specific page of interest. Moreover, prior knowledge about the data may help in determining some of the parameters like the default timeout period discussed earlier.

- ***Statistical Analysis***

  Descriptive statistical analysis (e.g. frequency, mean, variance...etc) can be performed on the session file variables such as page views, viewing time and length of navigational paths. The output of applying statistical methods could be determining the most frequently accessed pages, average viewing time of a page, average length of navigation paths to a specific page, or the most common invalid URI. Despite its lack of depth, the output of statistical analysis can occasionally help in reorganizing web content, making better marketing decisions and enhancing system performance and security.

- ***Sequential Pattern***

  Sequential pattern discovery mining [21] is concerned with finding inter-session patterns such that the presence of a set of items is followed by another item in a time-ordered set of sessions. Each access record in the log file includes the time of access, so that when preprocessing the data, these timestamps will be attached to their sessions. Discovering the sequential patterns allows the organization to predict the users' next click, which is vital information to tailor advertisements to that specific user or group. One example of sequential pattern discovery output could be that:

  "50% of users, who bought book1, also bought book2 after 10 days"

  This could mean that these two books are related (like being two parts of one novel), so offering a deal for both books could be worth consideration. Another example could be to find the common characteristics of all users interested in "Book1" in the period "May 1st- May 15th".

- ***Path Analysis***

  Path analysis methods [20] are concerned with representing the website as a graph, and then determining the most frequently visited paths. The most obvious graph is obtained by representing the physical layout of the website, where the pages are the nodes, and the hyperlinks as directed edges. Other graphs may represent the edges as the number of users going from one page to another or the similarity such as in [23], or using Markov Chains [25], or using special trie-like structures such as the WAP-tree [26]. An example of the output of path analysis could be:

  "75% of clients who accessed the web site started from the page '/company_news/'"

  This means that the page '*company_news*' is the first page that most visitors

access. Hence, making sure that it links to all other pages might be a good idea.

- ***Association Rule Mining***

  Association rule discovery [19, 24] is concerned with finding the associations among data items, where the presence of one item in a session implies (with a certain degree of confidence) the presence of other items. In the Web Usage Mining context, this refers to the set of pages accessed together by a single user. Note that these pages are not necessarily related via hyperlinks. An example of an association rule might be:

  "60% of users who accessed the mp3 player page, also accessed the sports page"

  This means that both pages are related, even though they probably are not linked via hyperlinks since they are in different departments, so this indicates the need to have hyperlinks between the two pages.

- ***Classification***

  Classification is the process of mapping a data item into one of several predefined classes [22]. In the Web Usage Mining domain, this means creating a set of profiles for users who have similar characteristics. Creating the profiles requires selecting the features that best describe the properties of a given class or category. These features could be demographical information such as age, location, sex...etc., or they could be access patterns. Classification is also called *Supervised Learning* since the classes are know in advance. An example of a classification rule could be:

  "70% of users who purchased an mp3 player are in the 18-25 age group and live in the East Coast"

  This rule suggests that creating a special offer for all users between 18-25 olds

who live in the East Coast might be a good plan.

- *Clustering*

  Clustering is the process of dividing data items into groups called clusters, where all the items in one cluster are more "similar" to each other than any other items in the other clusters. This is sometimes called *Unsupervised Learning*, since neither the number nor the characteristics are known in advance as in Classification. In the Web Usage Mining context, this means grouping all the users with similar characteristics (age, sex, access patterns [2, 6]...etc) in a single cluster. Clustering can facilitate the development and execution of future marketing strategies, such as dynamically changing the content of a website based on the users' behavior or demographic properties as in [5].

This thesis uses a modified version of a clustering algorithm called Hierarchical Unsupervised Niche Clustering (HUNC) algorithm [6], to discover evolving profiles. HUNC is discussed in Section 2.4.

2.3.3 Patterns Analysis

After discovering the interesting usage patterns on the web, they are analyzed with specialized tools to better understand them. The analysis tools are a mix of different fields including statistics, graphics, visualization and database querying.

Visualization can offer a successful mean to help people better understand and study various kinds of output. [18] developed the WebViz tool for visualizing web access patterns. The Web is represented as a directed graph where nodes are the pages and edges are the hyperlinks. The access patterns are used to formulate a web

path, and the user of the tool can analyze any portion of the web site, and see how users are moving from one page to another.

On-line Analytical Processing (OLAP) is a powerful tool that provides strategic analysis, using the data in data warehouses, for the purpose of aiding in meeting business objectives. Multi-dimensional data is represented as a data cube, and this allows the near instantaneous analysis and display of large amounts of data. Access logs can be seen as multi-dimensional and they grow rapidly over time.Hence, OLAP can be applied to better analyze and understand usage patterns. An SQL-like querying mechanism has been proposed for WEBMINER [19], and it provides a simple way to extract information about association rules. For example the query

```
SELECT association-rules(A*B*)
FROM log
WHERE date >= 01/01/07 and domain = "com" and support = 2.0
```

will extract all association rules in the ".com" domain, that are after Jan 1st 2007, have a support of 2 percent, and contain the URLs A and B. Other sophisticated pattern analysis tools have been proposed, such as MINT in [26].

2.4 HUNC

HUNC is a hierarchical version of the Unsupervised Niche Clustering (UNC) algorithm. UNC is an evolutionary approach to clustering proposed by Nasraoui and Krishnapuram in [14], that exploits the symbiosis between clusters resulting from web usage mining and genetic biological niches in nature. UNC uses a Genetic Algorithm (GA) [16] to evolve a population of candidate solutions (user profiles) through generations of competition and reproductions. UNC has proven to be robust to noise and makes no assumptions about the number of clusters. However, UNC was formulated based on an Euclidean metric space representation of the data.

Table 2.1: Log File Sample

IP Address	Time	Method/URL/Protocol	Status	Size	Agent
67.99.46.141	[01/Jan/2007:00:20:05 -0500]	"GET /favicon.ico HTTP/1.1"	404	296	"Mozilla/5.0 (compatible;Google Desktop)"
65.54.188.146	[01/Jan/2007:00:56:45 -0500]	"GET /robots.txt HTTP/1.0"	404	283	"msnbot/1.0 (+http://search.msn.com/ msnbot.htm)"
65.54.188.146	[01/Jan/2007:00:56:45 -0500]	"GET /arwild01/ HTTP/1.0"	404	292	"msnbot/1.0 (+http://search.msn.com/ msnbot.htm)"
65.11.237.191	[01/Jan/2007:01:01:15 -0500]	"GET /rjmil-lol/fm/txt/fm38599.html HTTP/1.1"	404	314	"Mozilla/4.0 (compatible;MSIE 6.0; Windows NT 5.1; SV1)"
74.6.131.201	[01/Jan/2007:01:55:52 -0500]	"GET /industrial HTTP/1.0"	301	324	"Mozilla/5.0 (compatible;Yahoo! DES lurp;http:// help.yahoo.com/help/ us/ ysearch/slurp)"
74.6.131.201	[01/Jan/2007:01:55:53 -0500]	"GET /industrial/ HTTP/1.0"	200	618	"Mozilla/5.0(compatible;Yahoo!DE Slurp; http://help.yahoo.com/help /us/ysearch/slurp)"
74.6.131.201	[01/Jan/2007:01:55:59 -0500]	"GET /indus-trial/left2.htm HTTP/1.0"	200	8371	"Mozilla/5.0(compatible;Yahoo! DESlurp;http://help.yahoo.com/ help/us/ysearch/slurp)"

HUNC generates a hierarchy of clusters which gives more insight into the Web Mining process, and makes it more efficient in terms of speed. HUNC does not need to know the number of clusters in advance like in most clustering algorithms (e.g. KMeans), can provide profiles to match any desired level of detail, and requires no analytical derivation of the prototypes. Moreover, HUNC calculates the similarities between web pages based only on the user access patterns rather than content.

2.4.1 Preprocessing the web log file to extract user sessions

The access log file is the raw data used to discover the user patterns/profiles. Each visit to a web page by each user is stored as a log entry; each log entry consists of information about that particular access such as the access time, IP address, URL page, etc. Table 2.1 shows an example of such a file.

The first preprocessing step is cleaning the log file. This is done by removing any entry that is an outlier or doesn't contribute to the mining process. These entries include: result of an error (indicated by the error code), requests with method other than "GET", and record accesses to image files (suffix is .jpeg, .gif...etc).

The next step in preprocessing is mapping each URL on the server to a unique number j in $1....N_u$ where N_u is the total number of valid URLs: the URLs are extracted from the log file, i.e. they are not hard-coded or known in advance. Then, user sessions are created. A user session is a series of requests originating from the same IP address within a predefined time period. The i^{th} user session is encoded as an N_u-dimensional binary vector s_j with the following property [2]:

$$s_j^{(i)} = \begin{cases} 1 & \text{if user accessed URL j during session i} \\ 0 & \text{otherwise} \end{cases}$$

The set of all sessions is denoted as S.

2.4.2 Assessing Web User Session Similarity

Dividing the sessions into clusters is done based on a similarity function that determines how much a session is similar to another. When this function is defined, the clustering process can be done by combining similar sessions into one cluster. The similarity measure in HUNC relies on two sub-measures [2]. The first measure is given by:

$$S_{1,kl} = \frac{\sum_{i=1}^{N_u} s_i^{(k)} s_i^{(l)}}{\sqrt{\sum_{i=1}^{N_u} s_i^{(k)}} \sqrt{\sum_{i=1}^{N_u} s_i^{(l)}}} \tag{2.1}$$

This is the cosine similarity between the two sessions. Looking at the nominator shows that the similarity increases as the number of common URLs increases. The

denominator helps in normalizing the value so it will be between [0-1]. This measure reflects how similar the sessions are, but it completely ignores the hierarchical organization of the web site. For example, when the URL is *products productA* compared to *products productB*, it will result in zero similarity because they are different URLs, however they are in the same category (products), so they are similar.

This has motivated developing the second sub-measure which takes into account the syntactic representation of URLs into account. A similarity measure between two URLs was defined as follows [2]:

$$S_u(i,j) = min\left(1, \frac{|(p_i \cap p_j)|}{max(1, max(|p_i|, |p_j|) - 1)}\right) \tag{2.2}$$

Where p_i denotes the path traversed from the root node to the node representing the i^{th} URL, and $|p_i|$ represents the length of this path. The similarity on the session level is defined by correlating all the URL attributes and their similarities in two sessions as follows:

$$S_{2,kl} = \frac{\sum_{i=1}^{N_u}\sum_{j=1}^{N_u} s_i^{(k)} s_j^{(l)} S_u(i,j)}{\sum_{i=1}^{N_u} s_i^{(k)} \sum_{j=1}^{N_u} s_j^{(l)}} \tag{2.3}$$

The problem with this similarity measure is that it uses soft URL level similarities. If two sessions are identical then $S_{2,kl}$ simplifies to

$$S_{2,kl} = \frac{1}{\sum_{i=1}^{N_u} s_i^{(k)}} \tag{2.4}$$

which is a small value depending on the number of URLs accessed. Whereas $S_{1,kl}$ for the same sessions will be equal to 1. For this reason, a new similarity that takes advantage of both S_1 and S_2 is defined as follows [2]:

$$S_{kl} = max(S_{1,kl}, S_{2,kl}) \tag{2.5}$$

This similarity is mapped to a dissimilarity measure, for use in clustering [2]

$$d_s^2(k,l) = (1 - S_{kl})^2 \tag{2.6}$$

2.4.3 The HUNC Algorithm

HUNC [6] recursively repeats the execution of UNC at different levels of the hierarchy to get profiles at different levels, i.e. the profiles at level i have more details than the profiles at level i-1 .

HUNC stops running when some predefined threshold values are reached. Three threshold values are used:

1. *Maximum number of hierarchy levels* L_{max}: An arbitrary value which depends on the level of detail needed in the profiles.

2. *Minimum allowed cluster cardinality* N_{split} : This threshold value represents the minimum size allowed for a cluster, because a low cardinality cluster, i.e. cluster with small number of URLs, may not be interesting.

3. *Minimum allowed scale* σ_{split} : This threshold value represents the maximal variance or scale of the profile when deciding whether to split it at the next level. A large scale means that the profile is more diverse.

HUNC's pseudo code is presented in Algorithm 1.

Algorithm 1 HUNC

```
INPUT: sessions , L_max, N_split and σ_split
OUTPUT:-Distinct User profiles (a profile = set of URLs and scale σ_i)
       -Partition of the user sessions into clusters (each session is
assigned to closest profile)

START HUNC
  Encode binary session vectors;
  Set current resolution Level L = 1;
  Start by applying UNC to entire data set w/ small population size;
  //This results in cluster representatives P_i and corresponding scales σ_i
  Repeat recursively until L = L_max OR all cluster cardinalities N_i < N_split
  or all scales σ_i < σ_split
      Increment resolution level:  L = L + 1;
      For each parent cluster representative P_i found at Level (L-1):
          IF cluster cardinality N_i > N_split OR cluster scale σ_i > σ_split THEN
              Reapply UNC on only data records x_j assigned (i.e.  closest)
              to cluster representative P_i;
END HUNC
```

2.5 Evolving Profiles

Most research in web mining has focused primarily on developing applications based on data collected from the web usage logs, web structure, and web content for *a particular time frame*. The results of such applications would represent only the users' behavior, web content, and web structure at that time. However, the nature of the Web is dynamic in terms of user behavior, web structure, and web content. The changes occur either rapidly or slowly. For example, on news channels, the data changes every hour, whereas in an encyclopedia like Wikipedia most of the content does not change very rapidly.

Web usage data changes have captured a lot of attention because they reflect the user's perspective of the web as opposed to the creator's perspective, which makes web usage mining a more interesting field from a business perspective. The changes in web usage are the result of more than one factor, such as changes in web content and structure, social, economical, political changes, and other factors.

This thesis is concerned with developing a framework that captures the changing

behavior of users on the web over time, and represents these changes through a set of evolving profiles that are good representatives of the users' activities at any given time.

Although a lot of research has been conducted on Web Usage Mining, only very few efforts have studied the dynamic nature of users' behavior. [8] studied the evolving nature of the web, where the Web was represented as a directed graph. Each node represented a web page and the hyperlinks were represented by the edges. The graph was analyzed at three levels: single node which studied the single node properties across different time periods, Sub-graph which studied the properties and interaction of a set of nodes, and the whole graph level which studied the properties and interaction for all pages. A more general study in [13] addressed the issue of learning evolving concepts, which are concepts that gradually change over time. The approach used different methodologies to maintain a set of representative examples derived from past experience, and it used aging and forgetting mechanisms to manage the examples.

[7] Proposed a new scalable clustering methodology that is inspired by the natural immune system's ability to adapt to a dynamic environment. The web server played the role of the human body, and the incoming requests played the role of new viruses, while a Dynamic Immune Learning system played the role of the immune system that continuously performed an intelligent organization of the incoming noisy data into clusters. This system showed an ability to handle a dynamic environment, and still required modest memory and computational costs.

Koychev [11] presented a method for gradual forgetting, that assigned a weight to each training example according to its appearance over time. This weight was updated over time, so that when it reached a minimum threshold, it would be forgotten. Koychev applied this method on the drifting of users' interests. A feature selection mechanism is used to identify the important features to a user, i.e. an explicit user profile. Then a probabilistic approach used these profiles for recommendation. For-

getting weights were employed on the feature occurrence in time, which reflected the feature significance estimation. The experiments showed an improvement in the recency of user's profiles and in the recommendations.

[12] proposed a framework to deal with the changing nature of the Web: it repeatedly mined the web log data in new time periods, and tracked how the discovered profiles evolved, and then categorized this evolution based on predefined categories: birth (completely new trends), death (trends that have vanished), atavism (trends that disappear for a while, then reappear again), persistence (trends that reoccur in consecutive time periods), and volatility (trends that go through birth then death then re-birth throughout the time periods). Profiles were categorized by keeping all historic profiles, and then comparing the new profiles with the older ones. Each profile was assigned a scale that represents the amount of variance of the sessions around a cluster center. This scale was used to determine the boundary around each cluster, and thus determine if two profiles were compatible. After a set of new profiles have been discovered, they are compared against the historic profiles, and based on the compatibility between the profiles, they will be categorized. Moreover, [12] enriched the discovered user profiles with the following facets: Search queries submitted to a search engine before visiting the web site, inquiring companies of the users whose IP address is mapped in that particular session, and the Inquired companies who have been inquired about during the session in that particular profile. These facets may be used further in classifying the profiles, and may present potential businesses benefits. However, this approach did not actually *update* the profiles: it just *tracked* their evolution through different time periods. Moreover, the detailed information about the profile quality was not maintained.

In contrast to [12], the framework proposed in this thesis does *update* the profiles. Thus, *only* distinct sessions will undergo the pattern discovery process in subsequent time periods. It is therefore a more scalable approach. Also, during the updating, all

the common old URLs are emphasized by updating their relevance, while new URLs can also be added to the profile. Hence, looking at the updated profiles through different time periods can give more detailed information about how they have really evolved.

[9] studied the effect of session and document similarity measures on the mining process and on the interpretation of the mined patterns in the harsh restrictions imposed by the "you only get to see it once" constraint on stream data mining. The study also proposed a similarity measure that couples the advantages of both the coverage and precision measures. The study applied the stream mining algorithm TECNO-STREAMS from [17] in the context of mining evolving Web click-streams. The TECNO-STREAMS algorithm had the advantages of scalability, robustness and automatic scale estimation. New data is generated as a stream, and it is processed in just a single pass, while a stream synopsis is learned and evolved, consisting of a set of cluster representatives with additional properties such as spatial scale and age. The size of the synopsis is constrained, so as not to comprise too many clusters, and preference is given to more recent arrivals in the data stream in occupying synopsis nodes. [9] also presented an innovative strategy to evaluate the discovered trends using some special metrics, and a visualization method that showed the hits for high precision and high coverage over time.

2.6 Scalability

Scalability is concerned with the ability of a system or a process to handle a growing amount of work over time. In the context of web usage mining, scalability means the ability to discover or update the usage patterns of users based on new incoming web logs [7]. Since for some web sites such as Yahoo.com and Wikipedia.com the usage traffic is huge, making the size of log files extremely large, the scalability of any web usage mining algorithm is a necessity for many real-world situations.

The huge size of web usage data presents a real challenge to HUNC and other conventional clustering techniques because these methods assume that all the data can reside in main memory. The framework presented in this thesis handles the scalability issue of web usage mining in an efficient way. When a new web log file is available, it will be preprocessed and converted to a session file. Then the session file is compared against the existing profiles, and these profiles are updated accordingly. The updates are done based on how similar the session is or how accurately it is represented by some current profile. After the profiles are updated, only distinct sessions, i.e. sessions that are not close enough to any of the existing profiles, will undergo the pattern discovery process. Since the users' behavior changes gradually, unless the whole content of the website has changed, the distinct sessions will only form a limited portion of all the new sessions. As a result, HUNC will still be able to handle a large number of incoming web logs efficiently, hence becoming scalable.

In the scenario where even the log file is too big to fit in the memory to process, the proposed framework can split the file into a number of batches based on predefined criteria. Then each batch will go through the preprocessing phase, and the distinct sessions will be accumulated into one file for all the batches. Again, under the assumption that the users' behavior changes gradually, new profiles will be discovered only from these distinct sessions.

2.7 Summary

Chapter 2 introduced an overview of web usage mining, its methods, steps, and applications. Also, the sources of usage data and some of the issues related to obtaining them were discussed. The HUNC algorithm, which will be used in the proposed framework, was also presented. Chapter 2 presented the notation of web use profiles, and scalability was introduced as an important requirement for any technique aiming at analyzing real-world web usage mining data.

CHAPTER 3

METHODOLOGY

In chapter 2, we reviewed some of the research in the web mining field, and how most existing methods do not even address the notion of change. Even the few approaches that tried to capture and handle the changing nature of the web, were concerned with studying the changes done on users' behaviors over a period of time [12], but failed to utilize this change directly to maintain and develop the evolving profiles.

On the other hand, the approach proposed in this thesis discovers the usage patterns over different periods of time, and captures the changes made to the patterns. The old profiles' contents and properties are updated as new data arrives, and only the distinct web log data (i.e. data that is not close enough to any of the existing profiles) will undergo the data mining process, which makes this approach scalable and more efficient in terms of complexity and resource usage.

When updating a profile, the interesting URLs -that are common between this profile and the new user sessions- are emphasized by updating their relevance (how many users accessed this URL in relevance with the total number of users in this profile). Also the brand new URLs in the session are added to the profile (with the frequency of occurrence overall). Thus, the updated profile reflects more detailed information about how the users' behavior changed over time.

3.1 Overview of the Proposed Methodology

The web usage mining process is traditionally done in several steps with only few variations. The steps can be summarized below:

1. *Retrieve the users activities represented as log files stored on web servers.*

2. *Preprocess the log files to remove any irrelevant data.*

3. *Discover the usage patterns using a web usage mining algorithm .*

1. *Interpret the discovered patterns (and optionally use them for the ultimate purpose of mining, like a recommendation system).*

The traditional main steps above have been used to discover usage patterns within one specific period of time, but they can arguably be reapplied periodically on the web data to try to capture the changes in navigation patterns. However, there are some concerns using this approach.

- Reapplying the steps periodically can either be performed on the whole historic data including the newly coming logs, or only on the new log files. The former approach weakens the ability of new usage trends to be discovered because their weight would be too small compared to old trends which would have gained strength over time. The latter completely forgets all previous patterns which may not be reasonable or efficient.

- The second issue is scalability. Since a usage data stream can grow to be huge, trying to discover the new behaviors from the accumulated log files each time will require significant computational resources, and could even be impractical or impossible for websites with huge traffic.

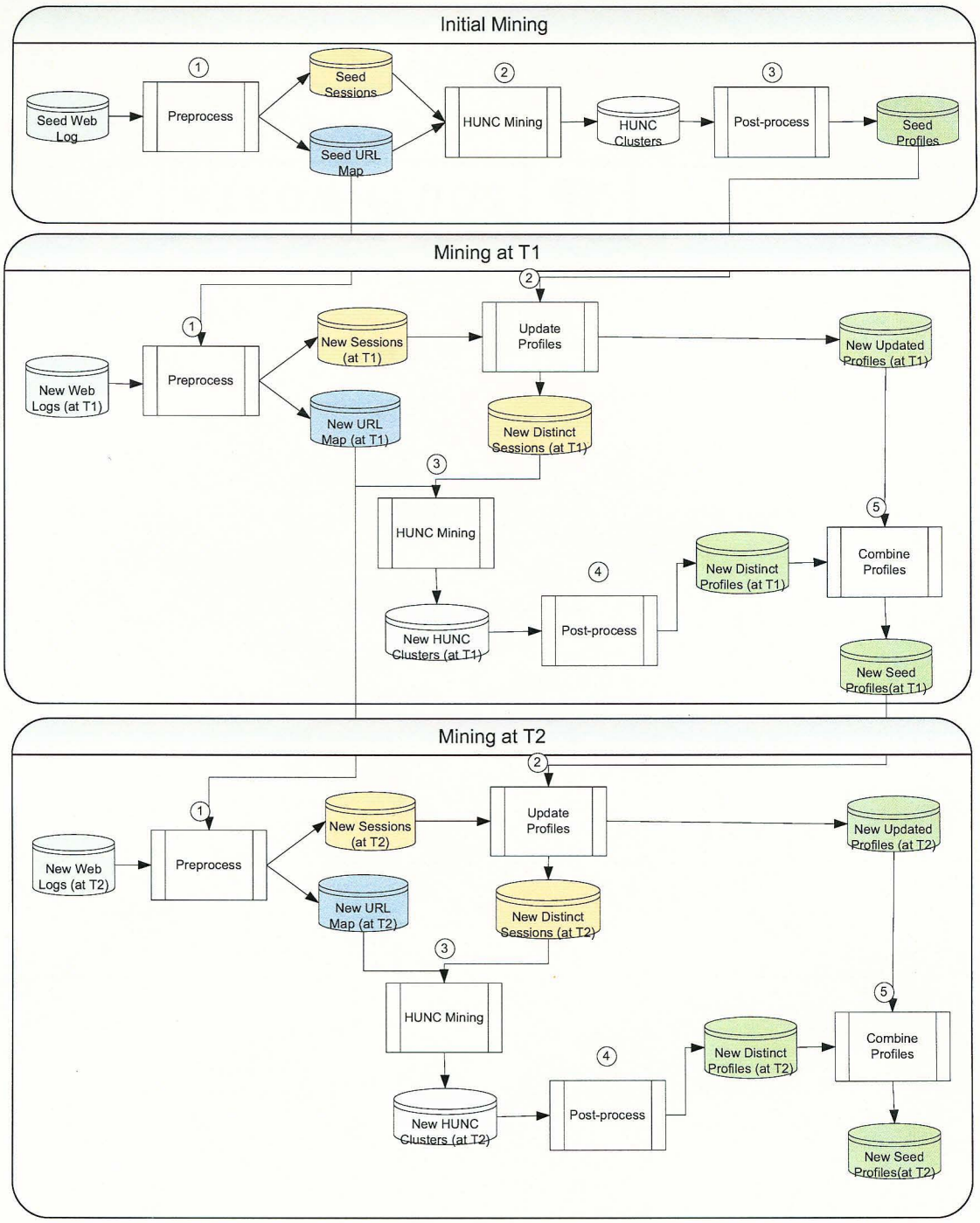
- The changes in the usage behaviors are not captured in detail, i.e. we do not know *how* users changed in their behavior. Even though the evolution of an entire profile might be monitored as in [12] and classified, knowing which URLs have changed or became more interesting was not enabled.

To overcome the above issues, this thesis adds two additional steps. The modified pattern discovery process is shown in Figure 3.1, and can be summarized, assuming that we start with a set of initial (seed) profiles mined from initial period, as follows:

1. *Preprocess the new web log data to extract the current user sessions.*
2. *Update the previous profiles based on the similarities with the extracted user sessions,*
3. *Re-apply clustering to the distinct user sessions (i.e. the ones not used in step 2 to update the previous profiles) using the Hierarchical Unsupervised Niche Clustering (HUNC) algorithm.*
4. *Post-process the distinct (new) profiles mined in step 3.*
5. *Combine the updated profiles with the distinct profiles to create the new seed profiles for future mining.*
6. *Interpret and evaluate the discovered profiles (and optionally use them for the main purpose of mining, like in a recommendation system).*
7. *Go back to step 1*

Figure 3.1 shows the pattern discovery process in three different time periods: the initial time, T1, and T2. The initial period is the first time the web logs are mined,

Figure 3.1: Proposed Pattern Discovery Process Flowchart



and since there are no historic profiles, all the sessions are considered distinct and they are all used to generate the first seed of profiles.

After the initial period, all the steps above are executed on subsequent time periods $T_1, T_2, T_3..etc.$ At each time period T_i, the profile seed from the previous period T_{i-1} is used in the updating process. Then the distinct profiles are created using the distinct data from T_i and combined with the updated profiles to create the profile seeds which will be used as the seed for time period T_{i+1}, and so on.

The URL file from T_{i-1} is used in T_i , which means that the list of all URLs is augmented each time that new URLs appear in the new data. The reason for this is to keep consistent indexing of all the URLs in the website throughout all the time periods.

3.2 Preprocessing the logs

During pre-processing, the web log files are cleaned by removing all irrelevant elements such as images, requests from search agents, and unsuccessful requests. Removing irrelevant accesses is necessary because these elements will affect the accuracy of the discovered patterns, and they will add an overhead to the computational and memory requirements.

Access requests for images are considered irrelevant because the images are typically embedded in a web page, and every time that the web page is requested, these images are requested automatically. Hence, these images do not represent the user's browsing behavior or interests.

Search engines send their web crawlers periodically all over the web to obtain information about web sites, and to index this information to be used in searching. Hence, a web log file will contain many requests from these web crawlers, and these

entries should be removed since they do not represent an actual Human browsing behavior. A heuristic for removing these requests is by looking for a set of arbitrary keywords like in the "Agent" field, such as "bot" to recognize these requests, or by observing the time difference between subsequent requests. A small time period like 1 second means that the user did not even view the page, so it is most likely a crawler.

After removing irrelevant entries, the page requests are grouped into units called sessions. Each session represents all the pages visited by a particular user within a predefined period of time. The sessions are represented as binary vector as shown in section 2.4.1.

Moreover, a URL index is created that includes all the URLs accessed in the web logs. This index is kept through future pattern discoveries and is always updated to reflect new URLs in the web site. Finally, a matrix of all URL-to-URL similarities, based on Eq. (2.2), is kept. This matrix is used when clustering the sessions.

3.3 Updating profiles

During this step, the new usage sessions will be used to update the old profiles, and only the distinct sessions will undergo the next step of discovering the new profiles/trends. The old profiles contribute to the evolving profiles from the last pattern discovery runs. The distinct sessions are the sessions extracted during preprocessing, and which were not used in updating old profiles because they were not found to be "similar" enough to any profile.

Before explaining how profiles are updated, we describe some important properties that describe profiles and sessions. A profile P_i is a vector representation of the cluster X_i, which is a set of user sessions that are more similar to each other than to the sessions in other clusters. The profile is represented as $P_i = (P_{i1}, .., P_{iN_u})^t$ where P_{ij} is the relevance weight of URL_j in cluster i, and is estimated by the conditional

probability of accessing URL_j during the sessions assigned to the cluster X_i

$$P_{ij} = p(s_j^{(k)} = 1 | s_j^{(k)} \in X_i) = \frac{|X_{ij}|}{|X_i|} \tag{3.1}$$

$$X_{ij} = \{s^{(k)} \in X_i | s_j^{(k)} > 0\} \tag{3.2}$$

$|X_{ij}|$ is denoted as N_{ij} in the following discussion. The relevance weight determines how much a URL is significant to a profile.

A profile P_i has a cardinality N_i which is the number of sessions that are closest to the cluster X_i , i.e. $N_i = |X_i|$. The cardinality does not necessary reflect the number of users represented by the profile P_i , since a user can have more than one session assigned to the same profile, which means that profile P_i represents the behavior of that specific user. The equation above can be re-written as:

$$P_{ij} = \frac{N_{ij}}{N_i} \tag{3.3}$$

Profile P_i has a scale measure σ_i^2 that determines how much the sessions in cluster X_i are dispersed around the cluster representative. The scale measure can be found using

$$\sigma_i^2 = \frac{\sum_{j=1}^{N} w_{ij} d_{ij}^2}{\sum_{j=1}^{N} w_{ij}} \tag{3.4}$$

Where w_{ij} is a robust weight that measures how typical a session x_j is in the cluster X_i.

$$w_{ij} = e^{-\frac{d_{ij}^2}{2\sigma_i^2}} \tag{3.5}$$

where d_{ij}^2 is the distance between session x_j and the cluster center for X_i as given by Eq.(3.7).

The scale measure σ_i^2 can be seen as a radius of the profile, and the more similar

the input sessions are to the profile, the smaller σ_i^2. Moreover, based on the robust weight w_{ij} it is possible to detect outliers which will have small weights, thus offering a means of distinguishing between good data and noise.

The similarity between a session S and a profile P can be assessed using the Cosine Similarity between two sets as follows:

$$Sim_{cos}(P, S) = \frac{|P \cap S|}{\sqrt{|P| \cdot |S|}} \tag{3.6}$$

Where |P| is the number of URLs in profile P, and |S| is the number of URLs in session S. The cosine similarity increases as the profile and the session share more common URLs, and it is normalized to be in the range [0-1].

The similarity between sessions S and profile P is mapped to the dissimilarity or distance:

$$d_{S,P}^2 = (1 - Sim_{cos}(P, S))^2 \tag{3.7}$$

Another measure of similarity called Robust Weight Similarity, will be used in the experiments, where we use the weight of a session defined in Eq.(3.5) with respect to the profile and compare it against a threshold value. The advantage of using the robust weight would be that the weights are normalized by the scale of each cluster; hence, they are less sensitive to the threshold, and they also depend on the profile. Profiles with smaller scales will be more strict in matching and vice vers for large-scale profiles.

One important issue when finding the similarity between a profile and a session is to consider only *significant* common URLs. The URLs in the profile are thus first compared to a threshold value P_{min}, and only URLs that pass this threshold values will be considered toward the count of similar URLs that will be used in finding the Cosine Similarity or the Robust Weight Similarity. Thresholding URLs is necessary to filter out the effect of potential outliers that might divert the profile from its accurate

and natural development. However, a too strict thresholding might cause discarding important updates to profiles in their infancy. Hence, choosing the right threshold value is vital. Updating the profiles can be summarized in algorithm 2.

`Update_Profiles` compares each session with all the old profiles, using the similarity between the session and each profile. The closest profile is chosen and its similarity is compared to a threshold value Sim_{min} and, if it exceeds this threshold value, then it will be updated. Otherwise the session will be classified as *distinct.*

Only the closest profile is updated in this approach, even though more than one profile might be close to the session. This approach is called the hard or crisp updating, where the session is a member of only one profile. An alternative approach based on soft or fuzzy memberships could allow a session to be a member of more than one profile. In this case, all profiles that are close to the session should be updated. There is a trade-off depending on which membership to use. A soft membership is expected to cause the coverage to increase, since the session will be represented in more than one profile. However, a higher coverage will almost always cause a lower precision, since the profiles would become more general and thus less detailed or accurate. A hard membership represents higher precision and lower coverage. From a business perspective, a hard membership means that each user is mapped to exactly one profile, making more detailed information about his behavior available, whereas, a soft membership will map each user to more than one profile, making more cross-product recommendations possible, at the risk of losing detailed information about the user's behavior.

In the `UpdateProfile` procedure, first the cardinality of the profile is incremented by one because the profile is accepting an additional session. Second, the URLs in the session are compared to URLs in the profile, and if the URL from the session is in the profile, then its weight is incremented because this URL is now found in an additional session. If the URL from the session does not exist yet in the profile, then

Algorithm 2 Update_Profiles

Input: The set of all new sessions S_a , The seed profiles $P(N_{ij}, N_i, \sigma_i^2)$
Output: The set of distinct sessions S_d, The updated profiles seed P_u

```
1 Update Profiles (S_a, P)
2 {
3 For each session S in S_a
4     {
5     Compute Sim(S, P_k) for all current profiles P
6     Find P_k that is closest to S
7     If Sim(S, P_k) > Sim_min then
8        UpdateProfile(P_i, S)
9     Else
10        Add S to S_d
11    }
12 }
```

```
13 UpdateProfile (P_i, S)
14 {
15 For each URL_j in P_i
16     {
17     If URL_j in Session S then
18        P_ij = (N_ij+1)/(N_i+1)
19     Else
20        P_ij = N_ij/(N_i+1)
21     }
22 For each URL_k in S but not in P_i
23     {
24     Add URL_j to P_i
25     P_ik = 1/(N_i+1)
26     }
27 Compute d^2_new = (1 - Sim(P_i, S))^2
28 Update profile variance: σ^2_inew = (σ^2_iold N_i + d^2_new)/(N_i+1)
29 N_i = N_i + 1
30 Update the End-Date of P_i to the last date of last access in S_a
31 }
```

it should be added, and its weight will be initialized. This weight will be low at the beginning because this URL is only in one session, however it should increase with time if this URL is interesting. All the URLs already in the profile, but that do not exist in the session will maintain their weight, but this weight will decrease slightly since the profile cardinality has increased.

Each profile contains a starting date and ending date, which reflect the period of time that this profile covers. This is important to keep track of which profile is most recent and which one was not recently updated. During the updating process, only the ending date is updated to reflect the date of the last access in the web log being processed. These dates could be used further during maintenance to identify which profiles become obsolete and should be archived.

Finally, the variance of the profile should be updated. As shown in line 28 in algorithm 2, the variance is an incremental version of the ratio of sessions' distance to the profile divided by the profile cardinality. Since the profile acquired a new session now, the new session's distance (d^2_{new}) should be added to the variance. The weight of each new session is considered to be 1 (unlike the variance definition in Eq.(3.4)), since we are already restricting only very similar sessions to update the profile (thus outliers are eliminated).

The complexity of Algorithm 2 is

$$O(N_s * |P| * Max(|URL|_P, |URL|_S)) \tag{3.8}$$

where N_s is the number of new sessions, $|P|$ is the number of profiles, $|URL|_P$ is the maximum number of URLs in a profile, and $|URL|_S$ is the maximum number of URLs per sessions.

Furthermore, we exploit the fact that Web sessions are extremely sparse (typically $<$ 10 URLs per session) as well as profiles, especially when applying a threshold on

URL significance. Hence, the maximum number of URLs is typically < 10. Moreover, the number of profiles tends to be small because only strong profiles are mined at any period. In the cleaning process discussed later, obsolete profiles can be deleted as well. Thus, the number of sessions is what really affects the performance of the updating algorithm. However, an important thing to remember is that updating of the profiles will most likely be done offline, so that it does not add any overhead in real-time.

3.4 Discovering distinct profiles

After updating old profiles, the new user sessions are analyzed so that new usage patterns can be discovered. These new trends may represent a potential new interesting market niche, or a radical change in the current market needs and behavior.

After the old profiles have been updated, only distinct sessions will undergo the pattern discovery process. The HUNC algorithm described in Section 2.4, is used to discover the new patterns, and the output will be denoted as the *new profiles*.

The output of the pattern discovery process at period (t) is a set of new clusters or profiles that are represented as a set of URLs. Each cluster has a variance measure σ_i^2 , cardinality measure N_i, sum of weights, and a density or fitness. The variance σ_i^2 is defined in Eq.(3.4), the cardinality is the number of sessions assigned/closest to this cluster up to period (t) and is donated as N_i as described in Section 3.3. The Density f_i of Profile i is defined as:

$$f_i = \frac{\sum_{j=1}^{N_i} w_{ij}}{\sigma_i^2} \tag{3.9}$$

where w_{ij} given by Eq.(3.5).

Since the distinct sessions are only part of the original sessions, the HUNC run time and resource usage can be reduced, which increases the scalability of this approach.

Figure 3.2: Sample Profile

```
Profile:  9, Num.URLS: 11, Cardinality:  58
StartDate:  30/Jan/1998:17:30:33 , End Date:  04/Feb/1998:13:37:34 ,
Variance:  0.0857
{0.98 - /courses.html}
{0.98 - /courses100.html}
{0.96 - /courses_index.html}
{0.82 - /}
{0.74 - /cecs_computer.class}
{0.34 - /courses300.html}
{0.20 - /courses200.html}
{0.17 - /courses_webpg.html}
{0.12 - /~joshi/courses/cecs352}
{0.10 - /courses400.html}
{0.10 - /people.html}
```

3.5 Post-processing the distinct profiles

The purpose of post-processing is to formulate the discovered patterns in a way that is understandable by humans, and is usable by higher-level applications.

The post-processing phase is the same as in HUNC, and it will be applied only on the newly discovered clusters. During the post-processing phase, each URL is mapped to the closest cluster. The set of all URLs in the same cluster constitutes a profile. A relevance weight is calculated for each URL as in Eq.(3.5), and the profile variance and cardinality are calculated as well - which are the same as in the cluster.

A sample profile is shown in the Figure 2.7. This profile contains 11 URLs, represents 58 sessions, and reflects the web activity between Jan 14th 1998 and Feb 4th 1998. Note that the profile variance is relatively small (0.0857), which means that the URLs are very close to each other.

3.6 Combining profiles

In this phase, the updated profiles and the newly discovered profiles are combined into one set of profiles that will serve as the new seed of profiles for the next pattern

discovery period.

Also in this phase, statistics about the pattern discovery process are collected, such as the number of updated profiles, the number of newly discovered profiles, and the number of distinct sessions. These statistics help in monitoring the performance of the discovery process, and in analyzing the output. For example, they can answer questions such as:

"During which period did the browsing behavior change the most?"

"Did changing the content of the website in period T1 change the users' behavior?"

3.7 Interpreting the profiles

The main purpose of web usage mining is to discover interesting information from raw data in web access log files, and to utilize this information in meeting some business or information organization goals. Thus, after summarizing the browsing behavior of users in the form of profiles, these profiles could serve as the input to higher-level tools that can analyze the usage patterns and make conclusions that help in the decision making process. The analysis tools draw on a mix of different fields including statistics, graphics, visualization and database management. Visualization can offer a successful mean to help people better understand and study various kinds of output.

The proposed framework generates well-formulated profiles that contain information and statistics about the profile and its contents. Some of the information is descriptive and can be used directly in higher level application. This includes the start and end date of the profile which mark the time period that this profile represents, as well as the list of URLs in that profile. Other types of information are mathematical, and are necessary to put the profile in perspective with other profiles, and each URL in perspective with other URLs in the same profile. This information includes the profile variance, profile cardinality, and URL weights. These statistics

are normally used when trying to decide which profile best represents a specific user, and which URLs are the most significant.

There are also some other pieces of information that are gathered for each pattern discovery cycle, and used for a higher level analysis of the profile evolution and interaction. For each cycle, the following statistics are collected:

1. The number of distinct profiles created in this cycle, which reflects which period witnessed the most and least radical changes in user behaviors.

2. The number of static profiles , which are the profiles that did not get any new updates from the previous cycle. This might signify the presence of out-dated profiles (since no new session matched them).

3. The number of distinct sessions, which could be used in evaluating the scalability performance of the usage discovery process.

3.8 Profile Maintenance

During the updating process, a few properties could be changed in the profiles: New URLs might be added, existing URL weights may change, and profiles variances may get updated. However, over time, some of the profiles or the URLs may become obsolete and should be omitted for several reasons:

- *Profiles are too old* and they have not been updated for a long time, so they do not represent interesting and recent usage patterns.

- *URLs were removed from the web site*, so keeping the URLs might have caused directing the users to non-existing pages.

- *The URL relevance weight is too low*, and it is not improving. This means that

this URL is not gaining any interest from the users, which could be because it is old, not accessible, or it appeared only once and disappeared.

- *The profile represents only a small number of sessions*, which is not cost effective from a business intelligence perspective.
- *A profile is becoming too general*, as indicated by a very high variance. In this case, the profile no longer represents a set of very similar user sessions. Hence it is of poor quality.

The existence of these obsolete profiles and URLs may cause the output of the pattern discovery to be inaccurate and misleading, and may even result in the dissatisfaction of users, in case the profiles are used to compute recommendations (e.g. a user might be directed to a non-existing or unrelated page). Moreover, these obsolete profiles add to the overhead in computations during the updating process.

Profile maintenance should be conducted in order to get rid of all obsolete profiles, and to make sure that the profiles are accurate, compact and up to date. The maintenance could be done by checking the profiles against some threshold values for the profile variance and URL weight. Given a maximum profile threshold σ_{max} and a minimum URL weight P_{min}, Algorithm 3 can be used for maintenance.

The complexity of the maintenance algorithm is $O(|P| * Max(|URL|_{P_{all}}))$, where |P| is the total number of profiles, and $|URL|$ is the number of URLs in a profile. Both these values are relatively low. Besides the profile maintenance algorithm can be executed offline, so that maintenance does not add an overhead to computations.

One important thing to notice about maintaining profiles is that the URL weights are compared against the threshold value before comparing the profile variance against

Algorithm 3 Profiles Maintenance

```
1 Profiles Maintenance (P_all)
2 {
3   For each profile P_i in P_all
4      {
5      For each URL_j in P_i
6         {
7         If  P_ij < P_min Then
8           remove URL_j
9         }
10     Update σ_i
11     If σ_i > σ_max Then
12         archive P_i
13     }
14 }
```

the threshold. This is important since deleting URLs will change the variance. Hence, after these deletions, the profile could become more interesting.

Algorithm 3 is pretty straightforward. However some important issues need to be addressed. The first issue is what threshold value to use; while the second issue is deciding when to run the profile maintenance algorithm.

Choosing the right threshold value is usually done by trial and error, and is domain-dependent. The profile variance represents how much a profile is dispersed, i.e. how different the covered sessions are from this profile. The variance value is in the range [0-1], so a high variance might mean that this profile covers a wider set of sessions, e.g. the profile contains accesses to URLs from different departments in an on-line store. A high variance is not desirable since it might be a result of a diversion of sessions from the original profile seed during updates, or it could signify that this profile represents two clusters, and hence it should be split in 2 profiles. On the other hand, a lower variance means that this profile contains sessions that are very similar to the profile. This is more desirable since it means that the profile is from a very compact cluster/group of users that truly represent a homogeneous interest.

Hence, choosing a threshold depends on the high-level application that will utilize the profiles; a news website might prefer more detailed profiles (so the variance

threshold would be low). On the other hand, an on-line store might occasionally prefer more general profiles for more cross-products recommendation between different departments (so the profile variance threshold value would be higher).

Another observation is that profiles are archived and not deleted. This means that those profiles that become inactive (no more sessions are similar to them) can still be useful when conducting some analysis on the profiles evolution over a longer time, and could be compared with active profiles to identify any similarities that might indicate a behavior group that was lost for a while and then came back.

The URL weight represents how much that URL is significant in the profile. The weight value is in the range [0-1]; a higher weight means that it is more significant, i.e. more sessions accessed this URL. Again, choosing a right threshold value is a domain-dependent task, so if the recommendations are required to be compact and accurate like in recommending a book, then choosing a high threshold is desirable.

A more complex issue is deciding when to run the maintenance algorithm. During the process of updating profiles in section 3.3, some new URLs will be added to the existing profiles, and their weight will be set to a small value $\frac{1}{N_i+1}$. hence, the maintenance algorithm would most likely remove these newly added URLs.

Moreover, in the process of discovering new profiles in Section 3.4 completely new profiles are created, and their variance might not be desirable, which might cause the maintenance algorithm to remove these newly created profiles. However, in both scenarios above, the newly added URLs might gain more weight over time so they would become interesting, and the newly created profiles might change their variance to a desirable value given time. Therefore, the timing of maintenance is crucial since it might cause the loss of an interesting and premature profile or URL.

As in choosing the threshold values, choosing the right time to run the maintenance algorithm is not trivial. A heuristic rule is to run it periodically with enough time in between, so that the profiles and URLs will have enough time to develop and stabilize.

Even in the scenario where a URL is created just before the maintenance time, and even if it was deleted, then, if it was really interesting, it will be added again and might gain weight in the next cycle.

Choosing the right time period is domain-dependent. A news website would probably perform the mining on a daily or even hourly basis since new URLs are added every hour. On the other hand, an on-line store would probably run its maintenance every month, with new events (e.g. introducing new products), or when marketing campaigns are started.

3.9 Summary

The proposed methodology, for mining and tracking evolving user profiles, was presented in Chapter 3. The main steps of tracking evolving users' behavior were discussed. These are: preprocessing new usage data, updating current profiles based on the new usage data, discovering usage profiles only from distinct usage data, post-processing the newly discovered profiles, combining updated and discovered profiles, and interpreting and evaluating the final profiles output. Moreover, Chapter 3, proposed a maintenance algorithm that is necessary to keep the discovered profiles accurate, compact, and up to date.

CHAPTER 4

EXPERIMENTS

To evaluate the performance of the proposed framework, it was used to discover the usage patterns from real web sites. Two web sites were used: University of Missouri's CECS department's web site, and University of Louisville's library website (library.louisville.edu). For each web site, a set of evolving usage profiles were extracted, and their quality was evaluated using a set of metrics. This chapter starts by defining these evaluation metrics in Section 4.1, then discusses the different parameters that will be used and their configurations in Section 4.2. Section 4.3 will discuss the experimental configurations. The results for each web site are presented in Section 4.4 and Section 4.5.

4.1 Evaluation Metrics

The output of the knowledge discovery process is a set of profiles, each of which is a set of URLs with additional metrics. Some of these profiles can be evaluated manually, by visiting its URLs and trying to determine whether these links are related or might represent a plausible usage trend. However, this evaluation can be subjective since what makes one user go from one page to another might be different from other users. Besides, it can be hard or even outdated to visit each URL for large profiles especially once URLs change location and content.

Table 4.1: Evaluation Metrics Summary (t = time period index)

Evaluation Metric	Notation	Desired Value	Range
Profile Variance	σ_i^2	Low	[0-1]
Profile Cardinality	N_i^t	High	[1-N_s]
Matching Sessions	-	High	[1-N_s]
Profiles Count	-	High	[1-N_s]
Profiles Pair-wise Similarity	$Sim_{cos}(P_i, P_j)$	Low	[0-1]
Profile Density	D_i^t	High	[1-∞]

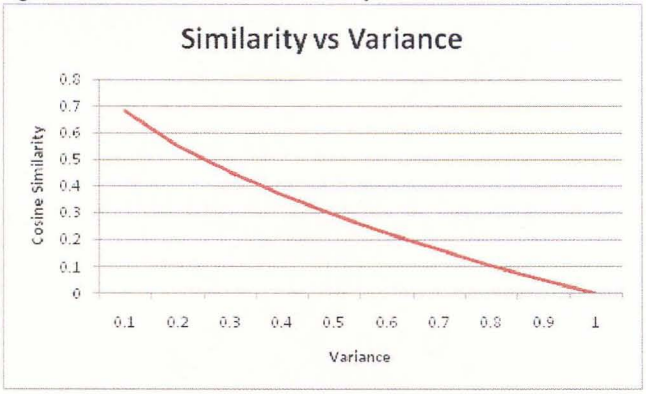
Also, it is not enough that the profile "descriptions" are plausible, since an additional criterion is that the profiles form good "clusters". A good clustering result is one, where data in the same cluster are very similar, while being dissimilar from data in other clusters. Another way to express this is that the clusters (or profiles) should be compact and separated. Thus, a more reliable and accurate method is to use a set of objective evaluation metrics to assess the quality of profiles. Table 4.1 lists the evaluation metrics and their expected ranges to be described in the next sections in detail (N_s is the number of sessions in the data).

4.1.1 Profile Variance

The profile or cluster variance (sigma) was defined in Eq.(3.4), and reflects how much the sessions in that cluster are dispersed. The profile variance is normalized in the range [0-1], and its value approaches 1 as the sessions in the corresponding cluster get further from each other (i.e. are less similar to each other). Hence, a lower profile variance is desirable, because it means that the sessions are closer to each other, i.e. the usage patterns for these sessions are similar, and this in turns attests to the quality of this profile.

The profile variance is proportionally related to the dissimilarity (difference) between the session and a profile. As the dissimilarity increases (i.e. sessions gets further from the cluster's core), the variance will increase. The dissimilarity was defined in Eq.(3.7). The cosine similarity can be redefined using the dissimilarity as:

Figure 4.1: Cosine Similarity vs. Profile Variance



$$Sim_{Cos} = 1 - \sqrt{d} \tag{4.1}$$

or it can be written in terms of the profile variance:

$$Sim_{Cos} = 1 - \sqrt{\sigma} \tag{4.2}$$

Figure 4.1 shows the relationship between the profile variance and the cosine similarity. A low variance will result in high similarity between the session and the profile, hence higher quality. For example, a profile variance value of less than 0.1 will result in a cosine similarity close to 0.7, which translates to the fact that the profile and the session share almost 70% of their content, which indicates a pretty good quality. A higher profile variance value like 0.5 will translate to only 30% of similarity between the profile and the session, which indicates lower quality.

4.1.2 Profile Cardinality

The profile cardinality is the number of sessions assigned to the corresponding cluster. A higher cardinality means that this profile is more popular and interesting. However a higher cardinality profile might also be a result of the "default" cluster

that captures all the "noise" in the data, which are sessions that are too different to form any strong and consistent usage patterns. When sessions are assigned to clusters based on similarity, many of these sessions end up with 0 similarity to all clusters (i.e. non-matching sessions), and end up lumped to the last cluster by default. As a result, this big cluster also ends up with a very high variance . The fewer "good" clusters that are discovered, the more non-matching sessions will fit this category and the higher the cardinality of this cluster. The cardinality will further be normalized to a percentage of the sum of all profile cardinalities, and will be used as a heuristic when comparing profiles variances, because comparing profile variances with close cardinality makes more sense.

When the profile gets updated, its cardinality is increased to reflect the number of new sessions that matched the profile, as shown in Algorithm 2. Hence, the cardinality of a profile at any given time is the sum of sessions that the profile acquired up to and including that time period. The cardinality N_i^t of a profile P_i at time period t can be written as follows

$$N_i^t = \sum_{j=1}^{t} n_i^j \tag{4.3}$$

where n_i^j is the number of sessions identified to be close to the profile i at time j. Furthermore, for evaluation purposes, the accumulated cardinality is normalized by the sum of all profile cardinalities at time period t. Thus, it can be defined as follows:

$$(N_i^t)_{norm} = \frac{N_i^t}{\sum_{j=1}^{|P|} N_j^t} \tag{4.4}$$

where $|P|$ is the total number of profiles.

4.1.3 The matching vs. distinct sessions

Capturing the changes in usage behavior is done by first updating the existing

profiles based on the matching sessions from the new logs, and then discovering the patterns from only the new (distinct) usage sessions. The *matching* sessions refer to the sessions that matched an existing profile and are used to update the properties of that profile, while the *distinct* sessions are the ones that are not close enough to any of the existing profiles, and therefore they are used to extract new profiles.

The percentage of matching sessions will represent how restrictive the discovery process was. A high matching percentage (low distinct percentage) might indicate that sessions that are not very similar were used to update the profiles, which can lower the quality of these profiles. However, this is not the case all the time, because a very high quality profile might also match a large number of very similar sessions simply because of the distribution of sessions. Moreover, a larger number of matching sessions affects the performance of the pattern discovery algorithm, since only a smaller part of the original log file will undergo the re-discovery process. This is one of the major advantages of the proposed framework over the traditional discovery process of all logs at once.

4.1.4 Number of profiles

After each discovery process, some profiles are created and some are updated. Tracking these numbers can give an insight about which time periods have witnessed changes in usage behavior, and what were the trends. Three types of profiles are defined:

1. *Discovered*: a profile that is completely new, and was generated from the "distinct" sessions during the current batch or period.

2. *Static*: a profile that has already existed from the last time period, but was not updated by any new sessions in the current period.

3. *Updated*: a profile that has already existed but was updated by the new "matching" sessions in the new period.

Tracking the number of profiles and their types over the evolution periods can reflect the overall usage trends of the users. A high number of discovered profiles may indicate that the usage patterns have drastically changed from previous times, which for example, would trigger the business need to capture these changes and develop marketing strategies to satisfy this new market.

A large number of static profiles might indicate that there was little change in the usage patterns, or that these profiles are becoming obsolete and no longer reflect the current activity. A large number of updated profiles is a good indication that the profiles are of good quality, since many sessions are still matching them.

However, making conclusions based only on the profile numbers and their types may be inaccurate, since the number of profiles and their types depend on the parameters used like the similarity threshold value. For example, a high threshold value would result in a larger number of discovered profiles and fewer updated profiles; whereas a lower value would increase the number of updated profiles.

4.1.5 Profile Pair-wise Similarity

A strong evaluation of the profile quality is to compare each profile with all other profiles, and determine if they are similar or not. A large number of very similar profiles indicates a poor clustering result, so our aim is a smaller number of similar profiles. The Binary Cosine Similarity defined in Eq.(3.6) will be used for comparing two profiles instead of a profile and a session. The similarity will be mapped to the dissimilarity measure defined in Eq.(3.7).

Since most profiles are not completely different from each other (they can share some URLs), a threshold value will be used to count the number of profiles that are too similar to each other. The value of 0.01 will be used as the threshold value, i.e.

if the difference between two profiles is less than 0.01 then the profiles are considered similar. The value of 0.01 in the difference means a similarity of 0.9 between profiles as given in Eq.(4.1).

4.1.6 Profile Density

The profile density is another quality metric for describing profiles, that was defined in Section 3.4 as the sum of session weights divided by the profile variance. The weight of each session is now considered to be 1 (unlike the variance definition in Eq.(3.4)), since we are already restricting only very similar sessions to update the profile (thus outliers are eliminated). Hence, the profile density (D_i^t) at time period t can be defined as follows:

$$D_i^t = \frac{N_i^t}{\sigma_i^2} \tag{4.5}$$

where N_i^t is the profile cardinality (i.e. the number of sessions that are assigned to or have updated this profile) up to time period (t).

The profile quality generally increases as its variance decreases, as discussed in Section 4.1.1, and a higher cardinality is also desirable as discussed in Section 4.1.2. Hence, the profile density D_i^t combines two quality metrics, and since it increases when cardinality is high and variance is low, then a high value for the density is desirable and indicates a high quality profile with high compactness (low variance) and more importance (high cardinality).

The advantage of using the density metric is that it considers two important quality metrics in combination. Using only the profile variance σ_i^2 , alone, to judge the profile quality is not sufficient, since a profile with very few similar sessions might have low variance, but that is not necessarily desirable if profiles are to represent mass user patterns. Moreover, using only the cardinality (N_i^t) is not accurate, because a

large cardinality might be a result of the "default" cluster which acquires all non-matching sessions regardless of their homogeneity.

4.2 Parameter configuration

The pattern discovery process depends on a number of parameters that affect the usage profiles and their quality. Varying the number and values of these parameters would help in determining the best configuration to be used, and help point out the weaknesses and strengths of the proposed framework. Only the most important parameters will be discussed in this thesis. Other parameters could be studied in the future.

4.2.1 Method of discovery

The first method of discovery is denoted as the "*Evolution*" mode, where different batches of data are processed and profile evolution is tracked through these periods. The second method is denoted as "*Traditional*" pattern discovery mode, which accepts all the web logs at once, and tries to discover the profiles in one shot. Therefore to emphasize the importance and scalability of the evolution mode discovery, the profiles discovered through "evolution" should be compared against traditional 1-shot profiles.

The profiles discovered in the traditional mode are expected to be of higher quality, since all log data are mined in one shot. Hence, each session will be compared to each other session during *all* time periods, and based on these similarities the profiles will be created. However, in the evolutionary mode, the session is only compared to all sessions in a *single* time period. The latter approach is more realistic since real website log data comes in batches with time (like data streams), and there is no way to know what the new logs would be.

4.2.2 Method of matching profiles

This parameter determines which similarity measure is used when comparing a new session to existing profiles, and deciding whether this session is close enough to this profile. The two methods that will be used are the *Binary Cosine Similarity* defined in Eq.(3.4), and the *Robust Weight Similarity* which is defined in Eq.(3.5).

The *Binary Cosine Similarity* depends primarily on the number of common URLs between the profile and the session, whereas the *Robust Weight Similarity* goes further by being more sensitive to the profile's variance as well. A profile with lower variance (typically indicating high quality) is more restrictive in matching, so as to maintain its quality.

4.2.3 Similarity threshold

For both methods of finding the similarity between a profile and new sessions, a threshold value (Sim_{min}), is used to control the strength of matching and thus the quality of resulting profiles. If the session is similar enough to a profile, i.e. the similarity is more than the threshold, then it will be used to update the profile (i.e. if $Sim(S_j, P_i) > Sim_{min}$ then update P_i).

Choosing the right threshold value (Sim_{min}) can be done by trial and error. A higher threshold value leads to more restrictive and higher quality profiles. However, if it is too high, then too many "similar" sessions will fail to match existing profiles and will be forced to contribute to the re-discovery of new profiles. This in turn would lead to "duplicate" or redundant profiles that keep getting re-discovered. Choosing the threshold value is also domain-dependent. For example, an on-line encyclopedia might prefer more accurate and restrictive profiles than an on-line store, since an encyclopedia aims to direct users to accurate and specialized sources of information, while the on-line store would prefer the users to browse through more products even though they may be less similar to the initial product that the user was looking at

(in hope of cross-selling).

For the Binary Cosine Similarity, two values are chosen: an average threshold of 0.3 and a more restrictive value of 0.5. For the Robust Weight Similarity, two values are also chosen: an average value of 0.3 and a more restrictive value of 0.6. These values were chosen based on some trial an error and historical results.

4.2.4 URL weight threshold

A URL significance weight threshold is used in two phases of the discovery process to make sure that only significant URLs are taken into account when comparing profiles. The first phase is in the post-processing phase in HUNC discussed in Section 3.5, where profiles are generated from the clusters discovered during the pattern discovery process in Section 3.4. The threshold value is applied to make sure that only URLs that are significant enough in the cluster are selected in the final profiles. This is needed to filter out any weak URLs which might risk affecting the profile description.

The second phase where the URL significance weight is used, is in the profile updating algorithm (Algorithm 2), where only significant URLs in the profile are compared to the current sessions to calculate the Binary Cosine Similarity or the Robust Weight Similarity. This is also necessary to make sure that an accurate and reliable update is done on profiles. Despite thresholding URLs when creating profiles, weak URLs might still find their way into the profile if they were part of a lengthy session with many URLs, that is used to update the profile.

However, the use of a URL weight threshold may introduce a critical trade-off, because on the one hand, the threshold will help prevent infrequent URLs from affecting the updating of profiles, but on the other hand, it risks discarding important updates to the profiles in their premature stage (i.e. while still weak). This is because URLs may start with low weight, particularly relative to the rest of the *well*

established URLs in the profile, and then might gain more weight over time, however using thresholding risks discarding them in the early steps.

As for the previous threshold values, choosing the right value is domain-dependent and can be done based on trial and error. Two URL weight threshold values are used in our experiments: an average value of 0.04 and a more restrictive value of 0.1 .

4.3 Experimental Configuration

The four parameters controlling the discovery of the usage patterns are shown in Table 4.2 with their different values. For the URL significance weight threshold, the term **URL_TH** will be used, for the Binary Cosine Similarity threshold, the term **CosSim_TH** is used, and for the Robust Weight Similarity threshold, the term **RobWT_TH** is used. Each parameter configuration will generate an experiment and its resulting evaluation metrics.

Table 4.2: Experimental Configuration

Method	Matching Criteria	URL_TH	CosSim_TH	RobWT_TH
Evolution	Binary Cosine	0.04	0.3	-
		0.04	0.5	-
		0.1	0.3	-
		0.1	0.5	-
	Robust Weight	0.04	-	0.3
		0.04	-	0.6
		0.1	-	0.3
		0.1	-	0.6
Traditional	-	-	-	-

The HUNC algorithm is used to discover the profiles from distinct sessions. Table 4.3 shows all the HUNC parameters and their values. Changing the values of these parameters might affect the resulting profiles quality. But they were chosen based

Table 4.3: HUNC Parameters

Parameter	Definition	Value
Min. Card.	The smallest allowable size of a cluster	20
Min. Card. to Split	If cardinality of a cluster is less than this, it will not be split	30
Var. Threshold to Split	If cluster's variance is less than this, it will not be split	0.1
Var. Factor	Factor of variance used as threshold for within niche distance in mating restriction	0.5
Min. Fitness Crossover	Threshold to identify valid individuals when restricting mating of good individuals from different niches	1000
Max. Num. of Levels	Maximum number of levels of hierarchical clustering	8
Population Size	The initial population size used in the genetic algorithm	20
Crossover Probability	The probability of performing crossover (per pair of individuals)	0.9
Mutation Probability	The probability of any bit of a chromosome being mutated	0.000005

on previous experiments on many website logs. Since this thesis aims to study the changing usage behavior over the web, and not the performance of HUNC, these parameter values will be the default values for all experiments.

4.4 Experiment 1: University of Missouri

4.4.1 The Dataset

This log data was collected from the CECS Department's website of the University of Missouri during a two-week period, from January 22nd 1998 until February 4th 1998. There were 32,770 access requests done during that period, grouped into 1704 sessions. To track the usage changes, the logs were divided into four batches:

- *Thursday, Jan 22nd - Sunday, Jan 25th :* 7,015 access requests (21.41%), 326 sessions (19.13%)
- *Monday, Jan 26th - Friday, Jan30:* 15,126 access requests (46.16%), 746 sessions (43.78%)
- *Saturday, Jan31-Sunday, Feb1:* 2,972 access requests (9.1%), 197 sessions (11.56%)
- *Monday, Feb 2nd - Wednesday, Feb 4th :* 7,657 access requests (23.33%), 435 sessions (25.53%)

The data was cleaned during the preprocessing phase as described in Section 2.4.1. Figure 4.2 shows the percentage of "bad access" and "good access" requests. Bad access requests include irrelevant and noise requests from search engines requests and requests that resulted in an erroneous status code. Good access requests are the remaining non-error generating requests.

Figure 4.2, shows the importance of the cleaning steps in the preprocessing phase, because all these bad requests would have adversely affected the profile discovery. Typically graphics requests may amount to 3-4 graphics per page or more (e.g. background picture, top banners, etc), that is why they end up being the majority of requests.

Figure 4.2: Missouri: Access Requests

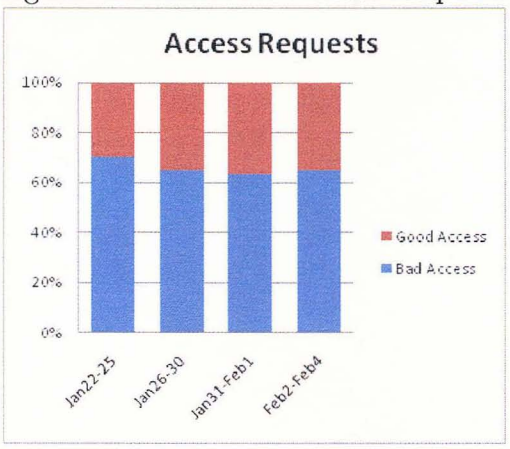


Table 4.4 shows the list of all profiles (after processing the last batch (Feb2-Feb4)) that resulted from mining this dataset. These profiles are the result of using Robust Weight Similarity with threshold value of 0.6, and URL threshold value of 0.1. The second column shows the total number of URLs in each profile, however for illustration purposes, only URLs with weight greater than 0.1 are shown. URLs were sorted in a descending order by the URL significance weight.

Table 4.4: Missouri: Final Profiles for RobWT (0.6) and URL TH(0.1)

Profile	Url#	Card	Variance	Content
0	22	63	0.1542	/courses_index.html, /courses100.html, /courses.html, /, /cecs_computer.class, /courses200.html, /courses300.html, /courses_webpg.html, /degrees.html,
1	29	68	0.230874	/cecs_computer.class, /people_index.html, /people.html, /faculty.html, /degrees.html, /grad_people.html, /staff.html, /degrees_grad.html,
2	40	46	0.268294	/~shi/cecs345, /~shi, /, /people.html, /people_index.html, /faculty.html, /~shi/cecs345/Lectures/04.html, /cecs_computer.class /~shi/cecs345/Projects/1.html,
3	231	223	0.592878	/~c697168/cecs227/left.html, /~c697168/cecs227, /~c697168/cecs227/head.html, /~c697168/cecs227/main.html, /~c697168/cecs227/handouts.html,
4	22	57	0.116559	/courses.html, /courses_index.html, /courses100.html, /, /cecs_computer.class, /people.html, /people_index.html, /faculty.html, /general.html, /index.html,
5	19	68	0.165445	/courses_index.html, /courses100.html, /courses.html, /, /cecs_computer.class, /courses300.html, /courses200.html, /courses_webpg.html, /courses400.html,
6	28	52	0.193953	/people_index.html, /people.html, /faculty.html, /, /cecs_computer.class, /grad_people.html, /staff.html, /undergrad_people.html, /research.html,
7	9	78	0.115836	/, /cecs_computer.class, /research.html, /~searc, /degrees.html, /general.html, /general_index.html, /facts.html,
8	13	88	0.079423	/~joshi/courses/cecs35, /~joshi/courses/cecs352/slides-index.html, /~joshi, /~joshi/courses/cecs352/environment.html, /~joshi/courses/cecs352/outline.html,
9	10	38	0.110454	/~joshi, /~joshi/sciag, /~joshi/research.html, /~joshi/sciag/logo.html, /~joshi/sciag/intro.html, /~joshi/resch/papers.html, /~joshi/dbrowse,
10	53	36	0.325243	/faculty/springer.html, /people_index.html, /faculty.html, /people.html, /faculty/keller.html, /faculty/chen.html, /faculty/plummer.html, /faculty/palani.html,
11	12	53	0.1091	/~shi/cecs345, /~shi/cecs345/java_examples, /~shi/cecs345/Lectures/06.html, /~shi/cecs345/Lectures/07.html, /~shi/cecs345/Lectures/05.html,
12	3	49	0.016306	/~yshang/CECS341.html, /~yshang/W98CECS345, /~yshang,
13	19	44	0.0902	/~saab/cecs333, /~saab/cecs333/private, /~saab/cecs333/private/lecture_programs, ~saab/cecs333/private/textbook_programs, /~saab/cecs333/final.html,
14	17	48	0.051113	/~c697168/cecs227, /~c697168/cecs227/main.html, /~c697168/cecs227/left.html, /~c697168/cecs227/head.html, /~c697168/cecs227/labs/main.html,
15	4	28	0.0218	/~manager/LAB/motif.html, /~manager/LAB/tin.html,
16	6	82	0.556638	/access, /access/details.html,
17	24	49	0.143885	/~saab/cecs333/private, /~saab/cecs333, /~saab/cecs333/private/assignments, /~saab/cecs333/private/lecture_programs,
18	17	61	0.2141	/cecs_computer.class, /courses.html, /courses100.html, /courses_index.html, /degrees.html, /degrees_undergrad.html, /degrees_index.html, /bsce.html,
19	30	110	0.708612	/~joshi/courses/cecs352, /~c697168/cecs227,
20	8	56	0.242	/~joshi/courses/cecs352, /~joshi, /~joshi/courses/cecs352/proj/overview.html, /~joshi/courses/cecs352/proj, /~joshi/courses/cecs352/slides-index.html, /~joshi/courses/cecs352/outline.html,
21	9	57	0.1266	/~shi/cecs345, /~shi/cecs345/java_examples, /~shi/cecs345/references.html, /~shi/cecs345/Lectures/07.html, /~shi, /~shi/cecs345/Lectures/09.html,
22	8	250	0.8325	/, /cecs_computer.class, /~saab/cecs333/private, /~saab/cecs333, /~saab/cecs333/private/assignments, /courses.html, /courses_index.html, /courses100.html,

4.4.2 Profile Variances

To study how the profiles variances change over time, and hence how their quality changes, the variance for each profile in each time period was plotted against time, in Fig.4.3 when using Binary Cosine Similarity to match profiles with sessions. A plot for each different configuration is used. The profile numbers in the legend are ordered (in a descending order) based on their cardinality percentage (i.e. the cardinality of the profile divided by the total cardinalities of all profiles at the last batch).

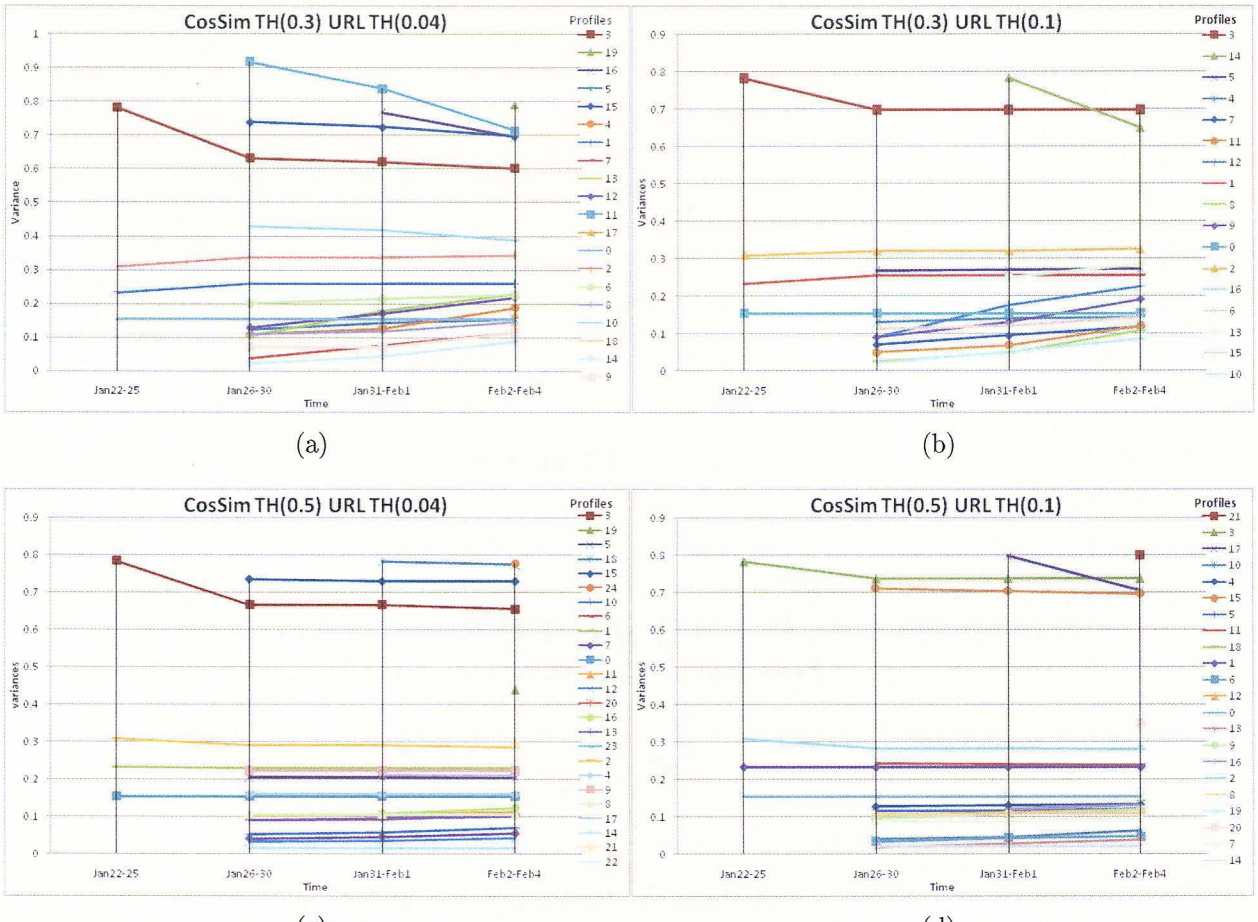
A low profile variance is desirable since it means that its assigned sessions are closer to the profile. Hence, if the profile variance decreases over time, then this profile can be considered to be improving and gaining quality. Fig.4.3(a) shows that profiles are generally decreasing in variance over time, especially the ones with higher cardinality (such as profiles 11, 15, and 3). This means that as these profiles capture more sessions, and they improve in quality over time. Profile number 19 seems to have high variance and high cardinality which is not desirable, however it was discovered only at the last time period, so over time, it might get improved. The low-cardinality profiles (like 7, 9 and 14) show an undesirable behavior, which is an increase in their variance, however these represent only a small portion of the sessions, so they are naturally sensitive to even small changes resulting from a few new sessions.

Fig.4.3(b) also shows a similar behavior to Fig.4.3(a) with the difference that the number of profiles with high cardinality and high variance is smaller. Thus, the majority of profiles have lower variance which is more desired. Thus, using a more restrictive URL weight threshold (0.1) before matching resulted in better quality profiles (as expected).

Figs.4.3(c) and (d) show similar behavior where the profile variances does not change much over time. However, the main difference was in the total number of profiles generated. Fig.4.3(c) has more profiles than Fig.4.3(d). So changing the URL threshold didn't affect the quality as much when using a more restrictive similarity

threshold of 0.5. The majority of profiles in Fig.4.3(c) and (d) have low variance which is good, but they are not improving over time, which means that these configurations are highly sensitive to the initial profiles, in contrast to the configurations in Figs.4.3(a) and (b) where variances did change over time. This is probably because the high similarity threshold makes it very hard for the new period's session to "match" the existing profiles. Thus they remain more stable compared to when lower similarity thresholds were used.

Figure 4.3: Missouri: Evolution of Profile Variances (Binary Cosine Similarity)



(a) (b)

(c) (d)

Fig.4.4 shows the profile variances over time when using the Robust Weight Similarity. Figs.4.4(a) and (b) show a low number of discovered profiles, low overall

variance, and stable variance over time. A low number of profiles means less diversity (i.e. more general profiles). So the change in the URL threshold did not seem to affect the variances, it only changed the number of discovered profiles. This seems to be a different behavior from using the Binary Cosine similarity where changing the URL threshold resulted in better quality profiles.

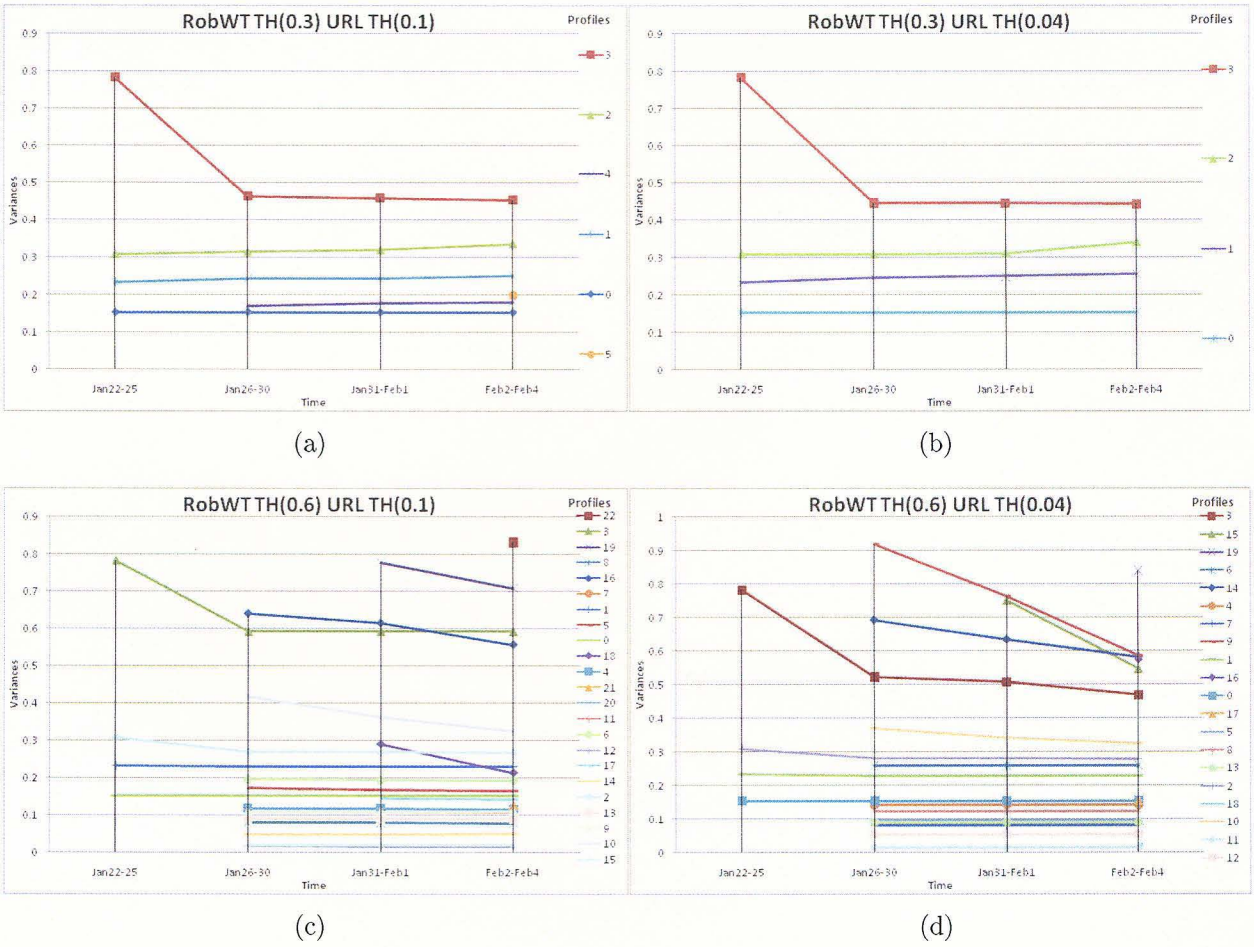
Figs.4.4(c) and (d) show a similar behavior to Figs.4.3(a) and (b), where the profile variances showed improvement over time. However, they have the advantage that even the lower cardinality profiles retain a constant variance over time. This desirable behavior is due to the fact that the Robust Weight Similarity is more sensitive to the profile variance, i.e. it is naturally more restrictive when the profile has lower variance, thus "shielding" even the vulnerable small profiles from noise.

Using a more strict Robust Weight Similarity threshold as shown in Figs.4.4(c) and (d) has resulted in more profiles than in Figs.4.4(a) and (b), i.e. more "detailed profiles". Another conclusion can be drawn from this figure is that the URL threshold value did not really affect the quality when using the robust weight threshold, whereas it had more effect when the cosine similarity was used. So based only on Fig.4.3and Fig.4.4, using the more restrictive Robust Weight threshold value has resulted in higher quality and detailed profiles. However we need to also look at the "rest" of the metrics like the inter-profile similarity before making a judgment.

To study the overall quality of the profiles, some aggregate metrics were calculated during all the time periods. Fig.4.5 shows the minimum, maximum, average, and median variances of the profiles at the end of evolution: Fig.4.5(a) shows the aggregates when using the Binary Cosine Similarity, while Fig.4.5(b) shows the aggregates for Robust Weight Similarity.

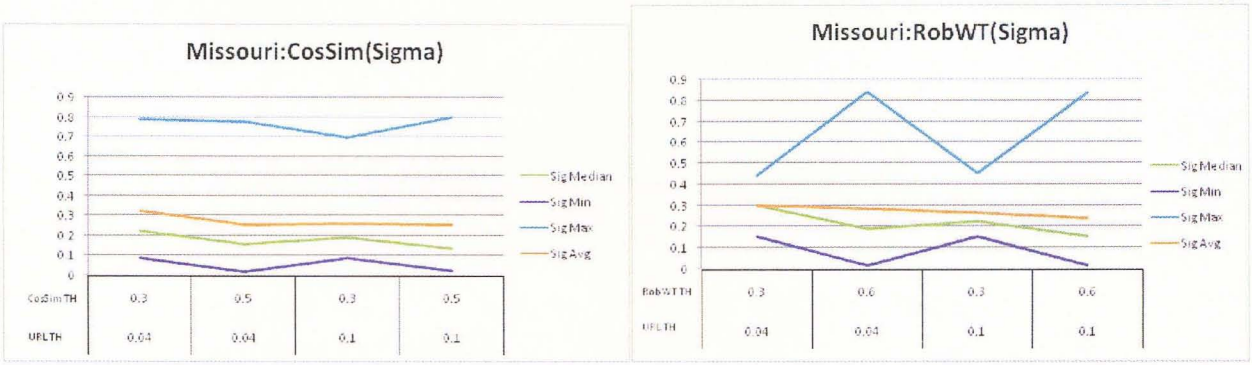
The overall trend regarding the aggregates in both charts seems to be similar, where the average and the median variances maintained similar levels. However the

Figure 4.4: Missouri: Evolution of Profile Variances (Robust Weight Similarity)



maximum variance decreased significantly when the similarity threshold was lower (0.3 in both CosSim and RobWT), and increased when the similarity threshold was stricter (i.e. using higher values of 0.5 in CosSim and 0.6 in RobWT for matching new sessions to old profiles). A general conclusion can be drawn that the URL weight threshold did not affect the aggregate variance metrics. In contrast to the maximum variance, the minimum, median and average variance seemed to decrease with stricter matching thresholds. Hence, a stricter matching results in an increase in the maximum variance and decrease in the minimum variance. This can be explained by the fact that stricter matching allows only "very" similar sessions to update most "good" profiles, resulting in profiles of better quality (lower variance). However, this

Figure 4.5: Missouri: Profiles Sigma Aggregates



(a) (b)

increased strictness will cause more new sessions to fail in matching most profiles and thus end up in the big "default" cluster that acquires all the heterogeneous sessions, which tends to have maximum variance, thus increasing its variance even more.

The difference between the two results in Fig.4.5 is in the value of the maximum variance; while the minimum, average, and median variances change very little (Fig.4.5(a) has a narrow range of values for the maximum variance (0.7-0.8) whereas Fig.4.5(b) has a wider range (0.4 - 0.8)). This means that using the Robust Weight similarity can result in a set of profiles where the maximum variance is around 0.4 (when the similarity threshold is 0.3), so all the profiles are essentially of good quality. However, going back to Fig.4.4, the profiles generated in this case were too general , i.e. only a few generic profiles were created, and they maintained their variance over time which makes them more sensitive to the initial profiles discovered.

4.4.3 Profile Evolution

The number and type of profiles discovered during the evolution gives an overall picture of the changes in behavior trends. *Distinct* or newly discovered profiles indicate a radical change in browsing behavior, while *static* profiles indicate low traffic

or obsolete profiles, and *updated* profiles indicate more stable representative profiles and/or similar browsing behavior by users.

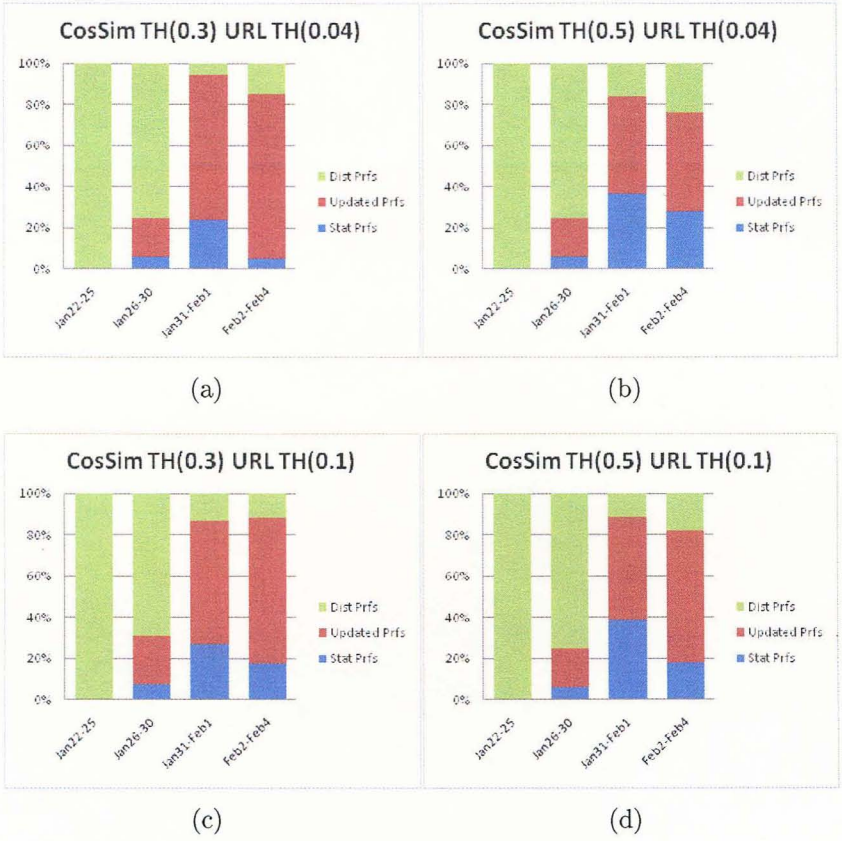
Fig.4.6 shows the percentages of each type of profiles for each time period during evolution when using the Binary Cosine Similarity for matching. The initial time period (period 1) always results in all distinct profiles since there are no profiles from previous time periods.

During period 2 (Jan26-Jan30), the profile percentages are almost the same for all configurations. Almost 80% of profiles are distinct which means that the usage behavior in this period may have changed by 80%. This might be due to many reasons: the initial data from Jan22-Jan25 was from Thursday to Sunday, with half of it on a weekend, so the traffic was low, and it might not be the best initial period to use. However, over time, profiles will evolve and eventually represent more typical everyday web usage patterns.

During the weekend in period 3 (Jan31 and Feb01), the traffic was the least, and the profile percentages were almost the same for two configurations Fig.4.6(b) and (d), where about 40% of the profiles remained unchanged because of low traffic, while 40% of the profiles got updated, and only 20% of the profiles were newly discovered. In contrast, the configuration in Fig.4.6(a) shows that about 20% of profiles are static, 75% got updated and 5% were newly discovered. The large number of updated profiles could be explained by the lower threshold value for similarity, which allows more sessions to be matched to existing profiles and cause more profiles to be updated. Fig.4.6(c) is similar to Fig.4.6(b) and (d) but with fewer static profiles.

At the end of evolution (period 4), the changes between different configurations became more apparent. Fig.4.6(a) has the highest percentage of updated profiles (about 80%) due to the lower similarity threshold (0.3). Fig.4.6(b) has more balanced profiles because it has a stricter similarity threshold. Fig.4.6(c) and (d) are similar to each other where the updated profiles are about 60%. Moving from Fig.4.6(c) to

Figure 4.6: Missouri: Profile Counts (Binary Cosine Similarity)



(d), the similarity threshold gets more restricted. However the percentage of updated profiles stayed the same which is the opposite of what happened with Fig.4.6(a). But this behavior can be explained by the more strict URL threshold, which ensures that higher quality profiles are generated, and hence even if the similarity threshold is loose, weak sessions will not be matched to these profiles.

Fig.4.7 shows the percentages of profile evolution when the Robust Weight Similarity was used. The profiles show more dynamic changes in profiles percentages compared to Fig.4.6that used the Binary Cosine Similarity.

Fig.4.7(a) shows that the percentages of profiles are the same for all time periods (except the initial time since they were all distinct). Using this configuration, only

four profiles were developed: one of them remained the same, while the other three were updated during all time periods. The low similarity threshold is the reason for having a low number of profiles, because a new session can easily be matched to any of the existing profiles, and hence old profiles get updated instead of new profiles being discovered. The variances for these profiles are low as shown in Fig.4.4, but they are more general, and sensitive to initial data.

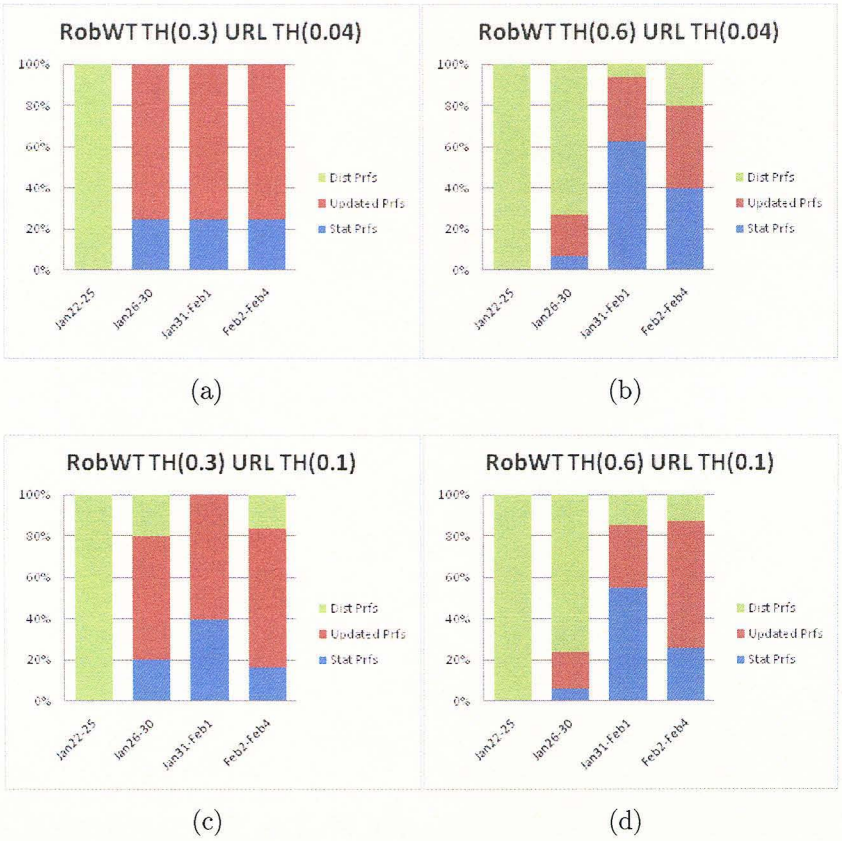
Fig.4.7(c) has also resulted in a low number of profiles (6 profiles), but some profiles were discovered during the evolution such as in period 2 (Jan26-Jan30). So even though the similarity threshold was low, the restricted (compared to Fig.4.7(a)) URL weight threshold caused some of the sessions to not match the existing profiles, and hence created more new distinct profiles.

Fig.4.7(b) and (d) are close in their behavior and they have many more profiles developed than previous configurations. During the second time period, about 75% of profiles were newly discovered, which is similar to Fig.4.6(b) and (d), and can also be explained for the same reasons , such as low traffic during weekend in the initial data. In the next time period more than 60% profiles remained static, which is also because that traffic was during the week end. The last time period witnessed more updated profiles and some static and distinct profiles, which is more typical of usage behavior and means that profiles are of good quality.

4.4.4 Matching vs. Distinct Sessions

The percentage of sessions that matched profiles to the sessions that were not close enough to any of the profiles is an important indication about how strict the matching process was. Fig.4.8 shows the percentage of matching sessions during all evolution period for both similarities matching methods (CosSim and RobWT). For

Figure 4.7: Missouri: Profile Evolution (Robust Weight Similarity)



(a) (b)

(c) (d)

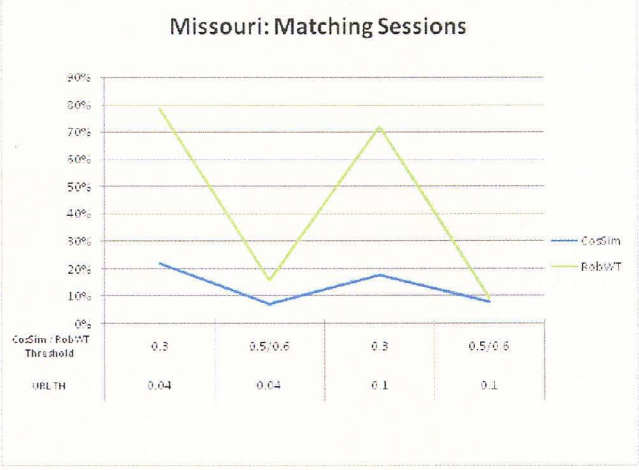
simplicity, the similarity threshold was joined on one axis for the two methods of matching.

A general trend for both methods is that the percentage of matched sessions is high when the similarity threshold is low (0.3), and is low when a stricter threshold is used (0.5 for CosSim and 0.6 for RobWT). This has an intuitive explanation, which is that a lower similarity threshold allows more sessions to be matched with existing profiles. Another thing to notice for both methods is that the URL threshold value had a minimum affect on the matching percentage: when using 0.1 as URL threshold the percentage is slightly decreased compared to using 0.04. So in general, the URL weight threshold is not significant when it comes to sessions matching percentage.

The RobWT method causes more changes in the percentage of matching sessions, when changing the similarity threshold, than the CosSim does. In fact, it goes from 80% when using 0.3 as threshold to only 15% when using a 0.6 threshold. In contrast, CosSim went from 20% when using 0.3 threshold to about 9% when using 0.5 threshold. This is also due to the sensitivity of RobWT to the profile variance. Since using both threshold values for RobWT result in overall low variance profiles as previously shown in Fig.4.4, these profiles would show more resistance to accepting different sessions, and hence, for a high similarity threshold of 0.6, they will not match new sessions as easily as the CosSim would.

A more detailed view of the distinct and matching sessions is seen in Fig.4.9, where the percentages of both the distinct and matching sessions are shown during all time periods for each parameter configuration, when using the Binary Cosine Similarity. It can easily be seen that the number of matching sessions is the highest in Fig.4.9(a) and (c), and less in Fig.4.9(b) and (d). This is due to the loose similarity threshold used in the former, and the strict similarity threshold in the latter. A higher threshold

Figure 4.8: Missouri: Matching Sessions Percentage

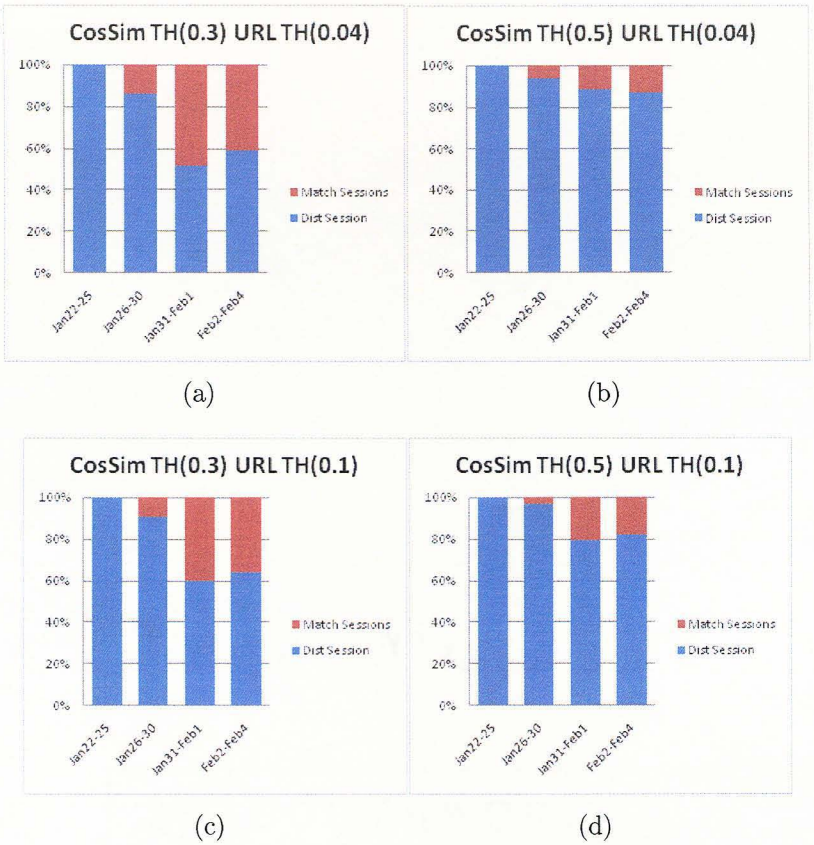


means that the sessions should really be closer to the profiles in order to update them, so fewer sessions would match profiles.

Another observation is that the time period 3 (Jan31-Feb1) witnessed the highest percentage of matching sessions relative to other time periods. This time period falls in a weekend, so the traffic in general is lower than week days so, given that good profiles have already been developed, the browsing behavior would match more profiles than it would in regular traffic.

Fig.4.10 shows the percentages of missing and matching sessions when using Robust Weight Similarity. Fig.4.10(a) and (c) are similar, and they show that the majority of sessions match the profiles and cause their update. Whereas in Fig.4.10(b) and (d), the majority of sessions did not match any of the profiles, which results in creating new profiles. The stricter similarity threshold (0.6) in Fig.4.10(b) and (d) makes it hard for new sessions to match the profiles, which in turn results in the high percentage of distinct sessions.

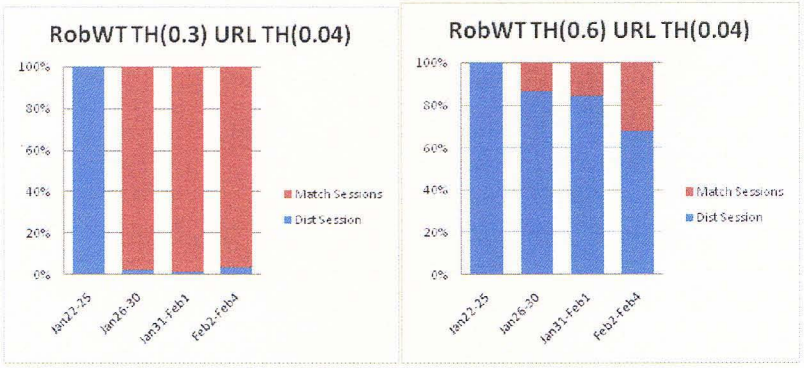
Figure 4.9: Missouri: Missing vs. Distinct Sessions (Binary Cosine Similarity)



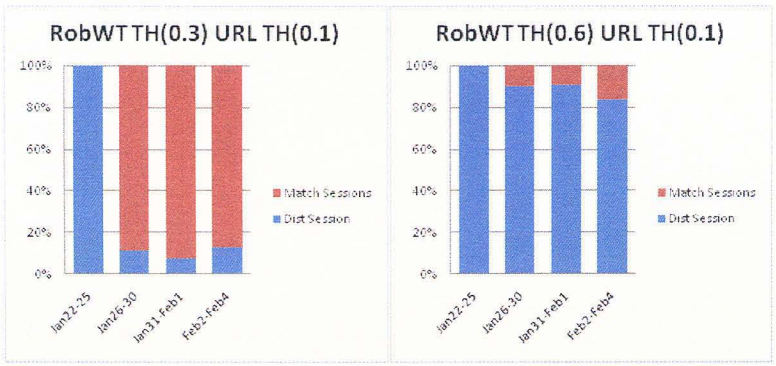
(a) (b)

(c) (d)

Figure 4.10: Missouri: Missing vs. Distinct Sessions (Robust Weight Similarity)



(a) (b)



(c) (d)

4.4.5 Profile Cardinality

The profile cardinality is the number of sessions that the profile represents. A higher cardinality means that the profile is more popular and represents more usage behavior. To compare the cardinalities of different profiles, the percentage of each profile's cardinality with respect to the total number of sessions is used.

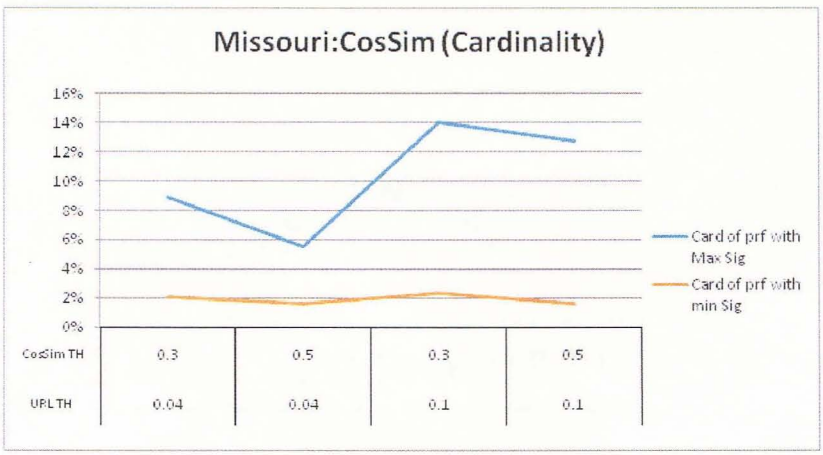
Fig.4.11 shows the cardinality percentage of both the profiles with maximum and minimum variance for the different parameter configurations. The cardinality percentage of the profile with minimum cardinality stays the same regardless of the different configurations. Recall that the profile with minimum variance is the highest quality (most compact) profile, so it is desirable to have it with high cardinality, however in this dataset its cardinality is low.

For the profile with maximum variance, the trends in both charts seem similar, where the cardinality percentage is higher when the similarity threshold is more restrictive. However, the Robust Weight threshold causes a more drastic change in the cardinality percentage than the Binary Cosine threshold. This can be explained by the fact that when the RobWT threshold is 0.3 only a few profiles are created, thus forcing one of them (the default profile) acquire all the sessions that do not match any profile.

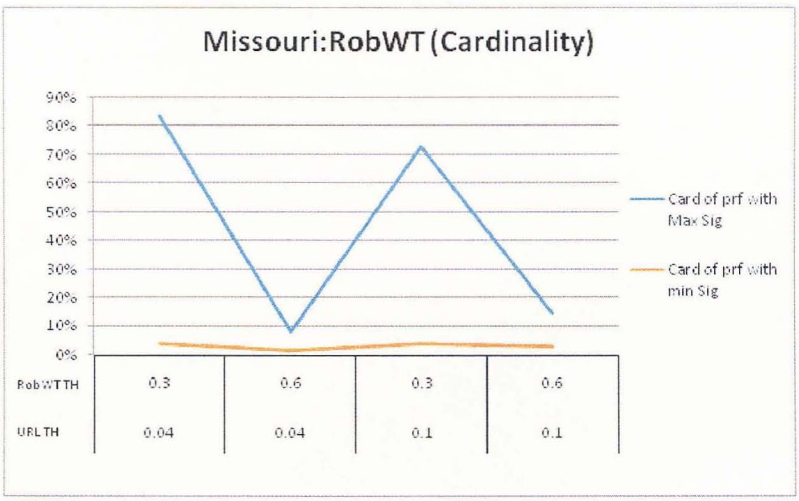
Another thing to notice is that in Fig.4.11(a), the cardinality percentage of the profile with maximum variance is higher when the URL weight threshold is higher (more restrictive), whereas in Fig.4.11(b), the URL weight threshold does not seem to have a big effect.

To show how the profiles are increasing or decreasing in their quality, the number of sessions assigned to the profile at each time period is tracked. Fig.4.12 shows the cardinality percentage of each profile at each time period with the Binary Cosine

Figure 4.11: Missouri: Cardinality of Max and Min Sigma

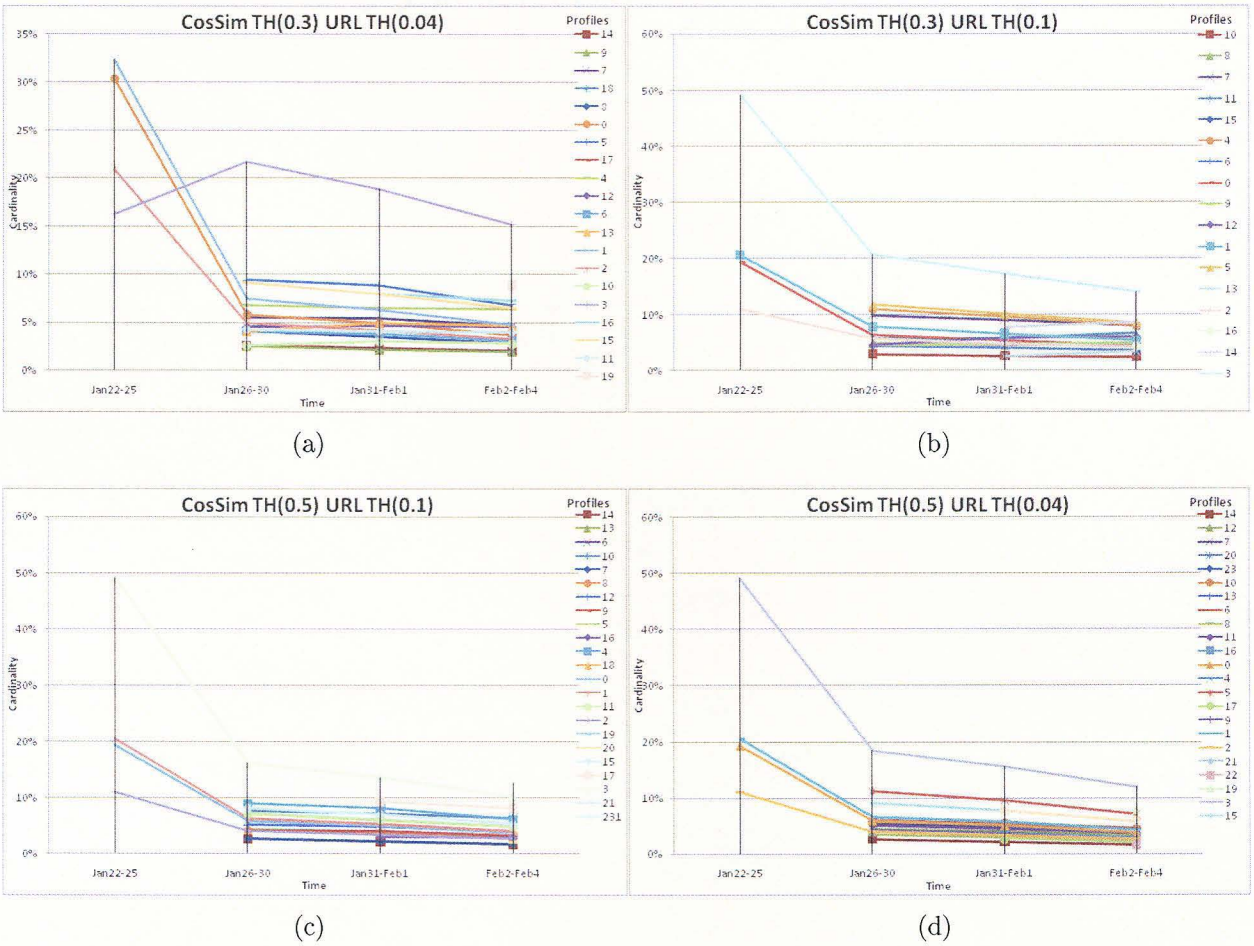


(a)



(b)

Figure 4.12: Missouri: Evolution of Profile Cardinality Percentages (CosSim)



(a) (b)

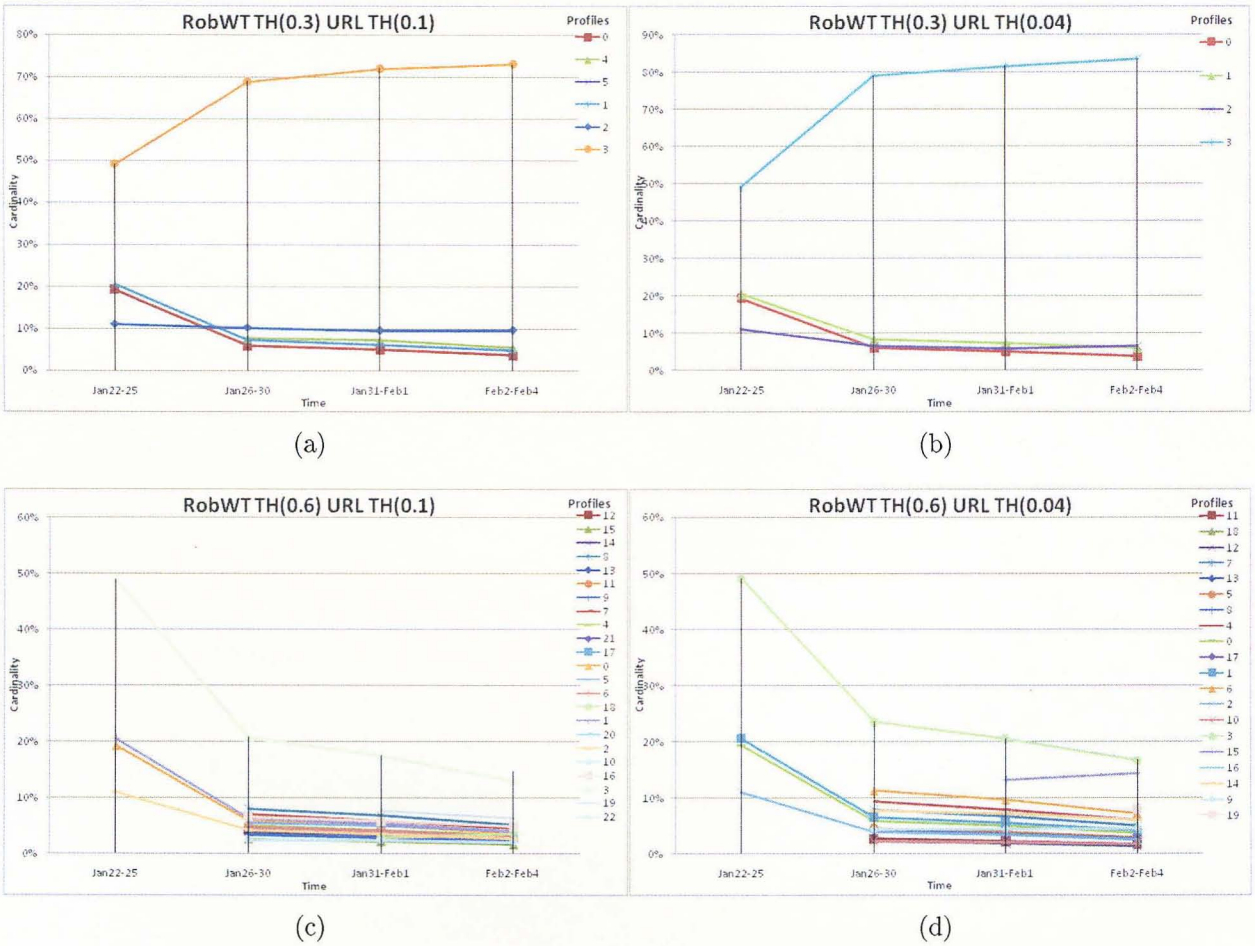
(c) (d)

Similarity. The cardinality of the profile at each time period represents all sessions matching this profile up to (and including) that time period. The cardinality is always normalized by dividing it by the sum of all profile cardinalities up to that time. The profiles are ordered (in a decreasing order) based on their variance.

All configurations in Fig.4.12 show similar behavior where the cardinality is decreasing slightly over time. The normalized cardinality is affected by the number of profiles discovered at each time period, since a large number of profiles means more competition on new sessions, and hence lower cardinality.

Fig.4.13 shows the profile cardinalities when using the Robust Weight Similarity.

Figure 4.13: Missouri: Evolution of Profile Cardinality Percentages (RobWT)

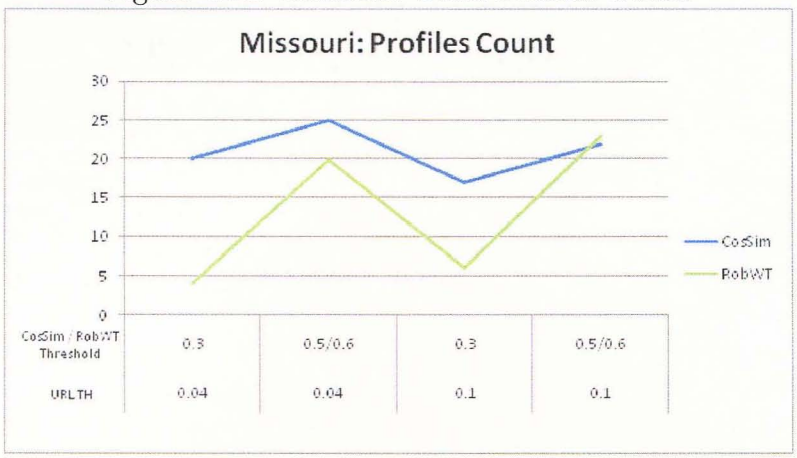


All configurations show that the majority of profiles are decreasing their normalized cardinality, except for two profiles in Fig.4.13(a) and (b). These two profiles do not have a lot of competition (because a low number of profiles were discovered), so they keep increasing their cardinality. However, both of them contain the highest variance which indicates that these profiles are of low quality, and most likely are the "default" profiles where the majority of new sessions are added.

4.4.6 Profile Count

The number of profiles generated at the end of evolution for each configuration is

Figure 4.14: Missouri: Total Profiles Count



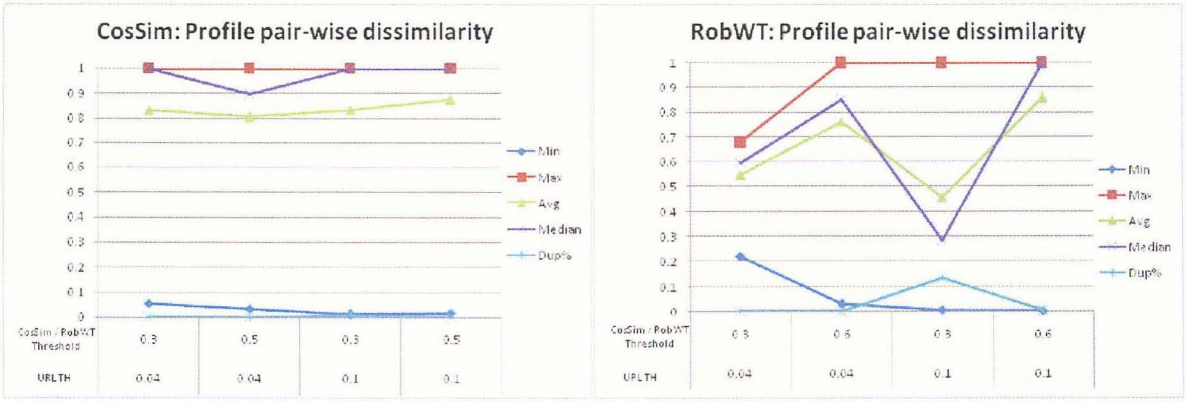
shown in Fig.4.14. A common trend between both matching methods (CosSim and RobWT) is that the number of profiles increases when a higher similarity threshold value is used. This is due to the fact that when fewer sessions succeed to match the current profiles (due to the higher matching similarity threshold), the distinct sessions would be forced to generate new (distinct) profiles. Another observation is that the URL weight threshold has minimal affect on the number of profiles for both matching methods.

4.4.7 Profile Pair-wise Similarity

A powerful profile discovery quality evaluation metric is to find how similar the profiles that result from the pattern discovery are to each other. If profiles are too similar to each other, then the updating algorithm was not able to match the new logs with the existing profiles accurately.

A difference matrix is created to find the pair-wise similarity between every two profiles at the end of evolution. The maximum, minimum, average, and median similarity is plotted in Fig.4.15, as well as the percentage of duplicate profiles. Duplicate profiles are defined as the ones with difference less than 0.01. This translates to pro-

Figure 4.15: Missouri: Profile Pair-wise Similarity Aggregates



(a) (b)

files who are at least 90% similar to each other (using binary cosine similarity), since $(1 - 0.9)^2 = 0.01$. See Section 4.1.1.

Fig.4.15(a) shows a desired behavior where the maximum, median, and average pairwise similarities are all greater than 0.8 which means that they are only 10% or less similar to each other. The minimum similarity aggregate is around zero which means that there are some profiles (at least one pair) that are very similar to each other. The duplicate percentage is also around zero which means that only very few number (if any) of the profiles are duplicates.

Fig.4.15(b) shows that when the similarity threshold is low (0.3) the median and average pairwise distances are low , i.e. most profiles are only 50% different, and the duplicate percentage is about 13% which is high compared to all other configurations. However, when the RobWT threshold value becomes more strict (0.6), the behavior improves and fewer profiles are similar to each other, resulting in fewer duplicates.

4.4.8 Profile Density

The profile density is a quality metric that combines the profile cardinality and variance. A high density is desirable since it means a high cardinality and low variance

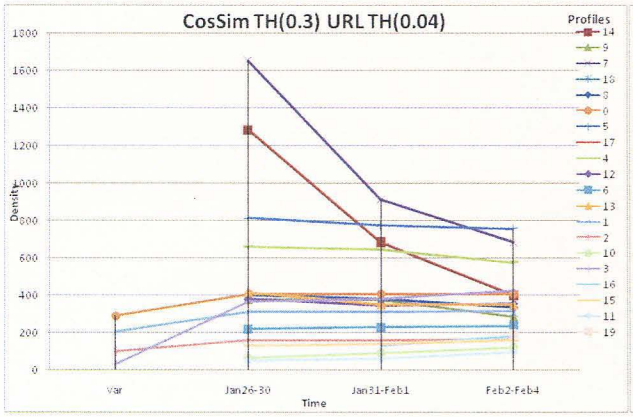
at the same time. The profile density is calculated at each time period by dividing the total number of sessions that belong to the profile until that time period (i.e. cardinality at T2 is the sum of cardinality at T1 and the new matching sessions in T1) by the variance of the profile at that time period. This means that a decrease in density indicate an increase in variance (since cardinality at time period T2 is always larger than or equal to T1).

Fig.4.16 shows the density for each profile at each different time period when using the Binary Cosine Similarity as the matching criterion. The profiles were ordered (in a decreasing order) based on their variance. For most configurations, the majority of profiles maintain their density over time, except for some profiles which decrease as in Fig.4.16(a) and (b). The decrease in the density means that sessions added to the profile are not very similar to old ones, and hence increasing the variance. A stable behavior like the majority in Fig.4.16(c) and (d) means that either the profile is not gaining any new sessions (out-dated) or the new sessions that are being added are not very similar to existing ones which increase the variance slightly.

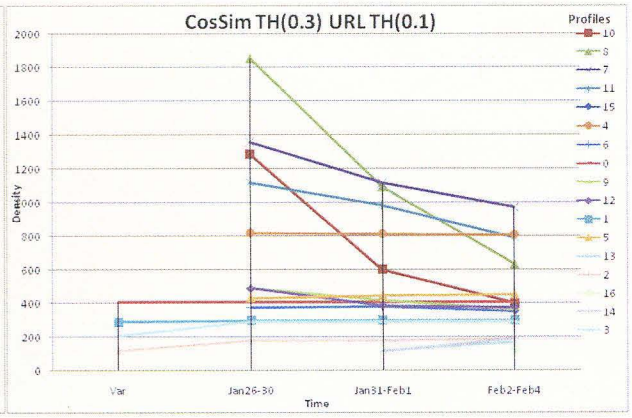
Fig.4.17 shows the profile densities when using the Robust Weight Similarity. Fig.4.17(a) and (b) show similar behavior were the profile with highest variance increases its density, and the rest of the profiles maintain a stable density over time. This means that the high cardinality profile is acquiring too many sessions (and that the increase in its variance is not enough to affect its density). This is an example of an undesirable high density, because this high-cardinality and high-variance profile seems to be the "default" cluster that all non-matching sessions are assigned to. This could not have been detected by looking only at the cardinality.

Fig.4.17(c) and (d) show more stable density over time (with some profiles increasing their density). Using the high similarity threshold have caused higher quality profiles to be discovered with a balanced ratio between their variance and cardinality.

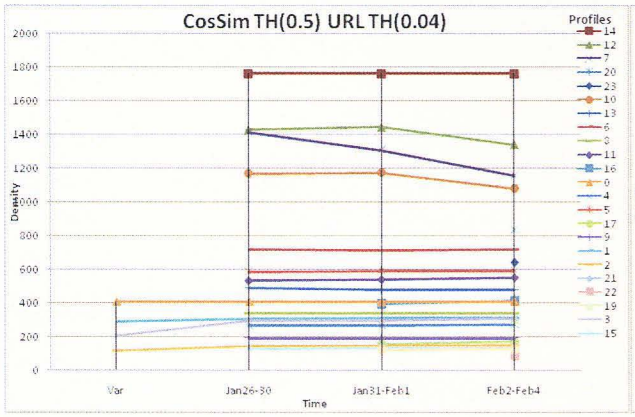
Figure 4.16: Missouri: Evolution of Profile Densities (CosSim)



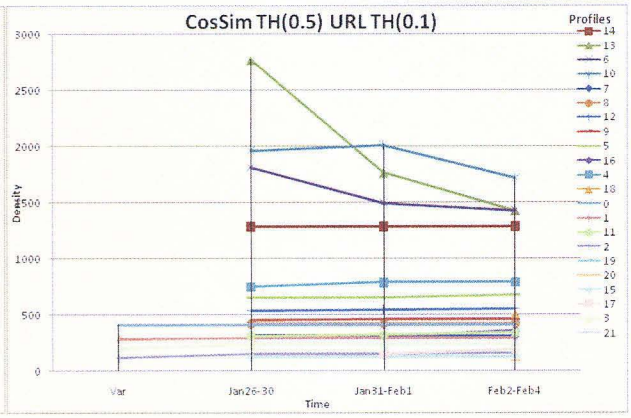
(a)



(b)

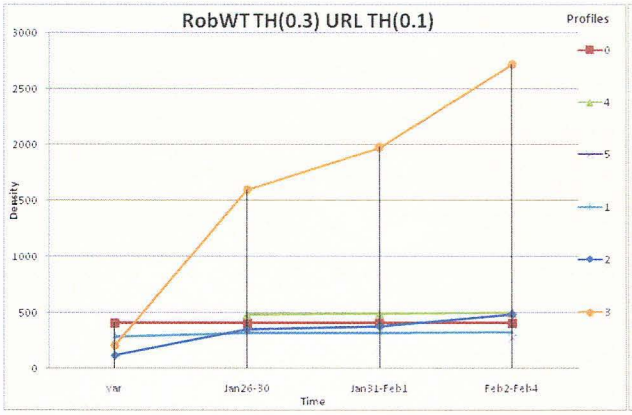


(c)

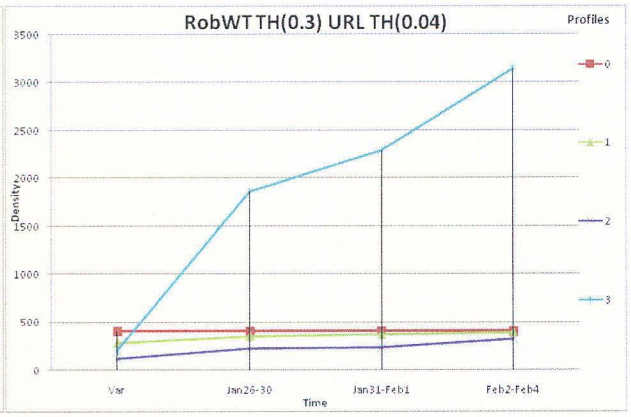


(d)

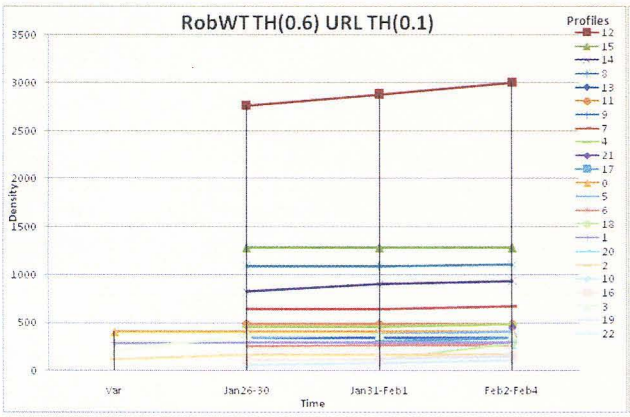
Figure 4.17: Missouri: Evolution of Profile Densities (RobWT)



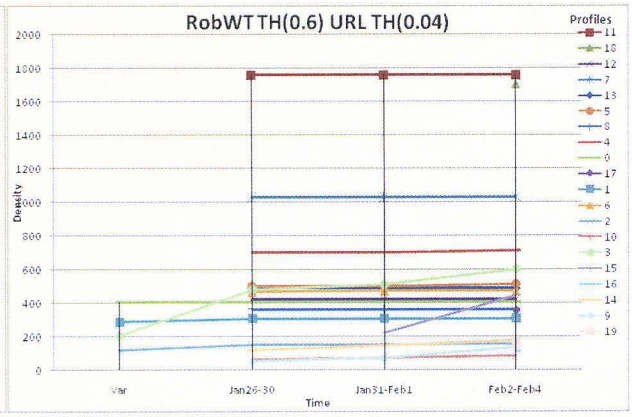
(a)



(b)



(c)



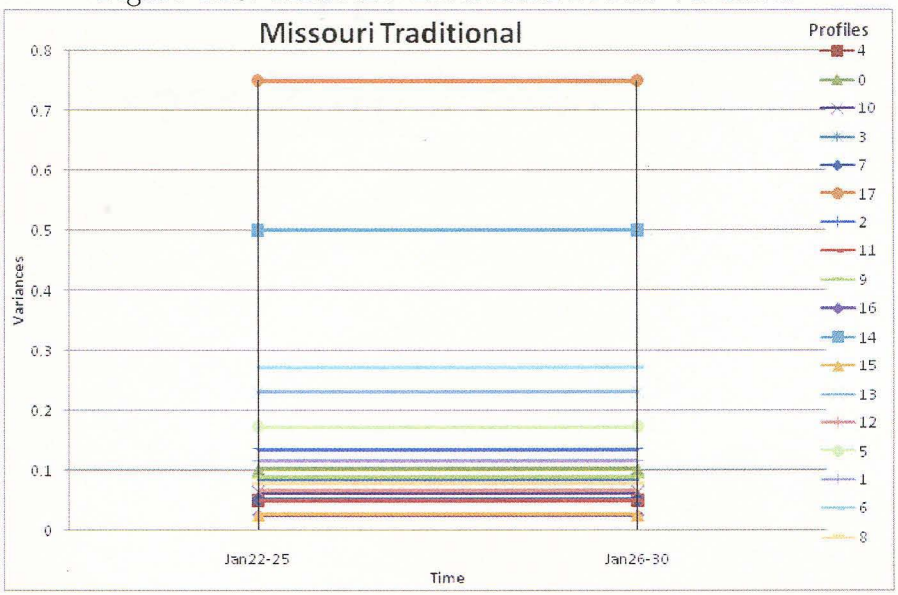
(d)

This consistent increase in densities may be the ultimate judge that matching using the Robust Weights (which depend on accurate variance estimates) yields the best quality profiles compared to Cosine Similarity matching.

4.4.9 Evolution vs. Traditional (Full run)

The last evaluation addresses the approach or *mode* of pattern discovery, which is either *evolutionary* (the focus of this thesis), where the pattern discovery is done in different time periods, or *traditional* where the full or entire usage data from all time periods is used to discover usage patterns in one shot.

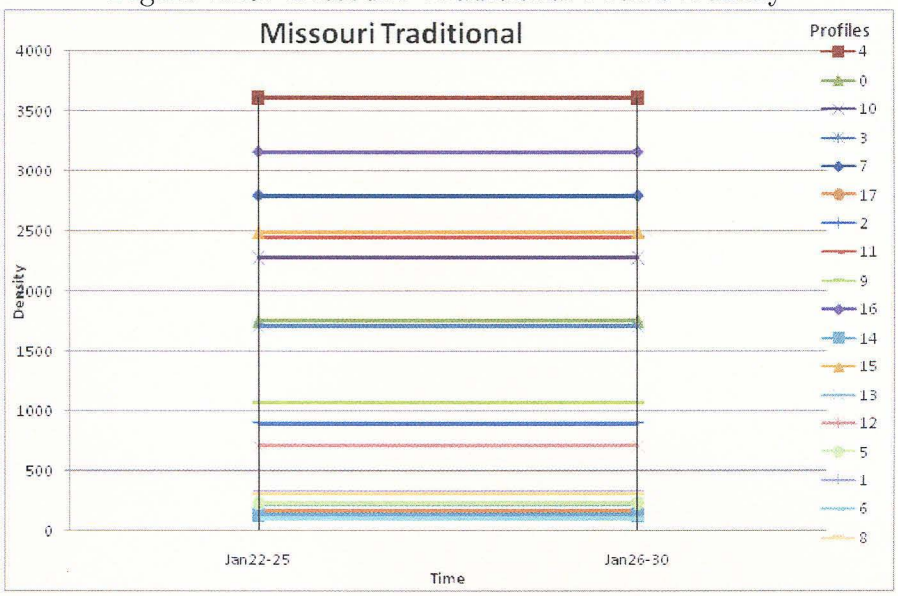
Figure 4.18: Missouri: Traditional Profile Variances



Profile variances for the traditional method are shown in Fig.4.18. The variances stay the same during all time periods (since they do not evolve), and they are shown at different times just for the purpose of illustration and for ease of comparison with Fig.4.3 and Fig.4.4 (for the evolution mode). The profiles were also ordered based on their cardinality percentage. The majority of profiles have a low variance, which means that most profiles from the traditional method are of good quality. These levels of quality were also found when using the evolutionary approach. However, we can conclude from Fig.4.4(d) that the Evolution mode was able to "refine" the profiles to lower variance (and hence improved compactness), by the end of all periods. The maximum variance was 0.6 in the evolution mode as opposed to 0.8 in the traditional mode (25% improvement).

Fig.4.19 shows the densities of the profiles discovered when using the traditional discovery approach. The density represents the ration between the profile cardinality and its variance. Profiles are order in a decreasing order based on their cardinality percentage. All profiles are of high quality since the density is high. Some of the

Figure 4.19: Missouri: Traditional Profile Density



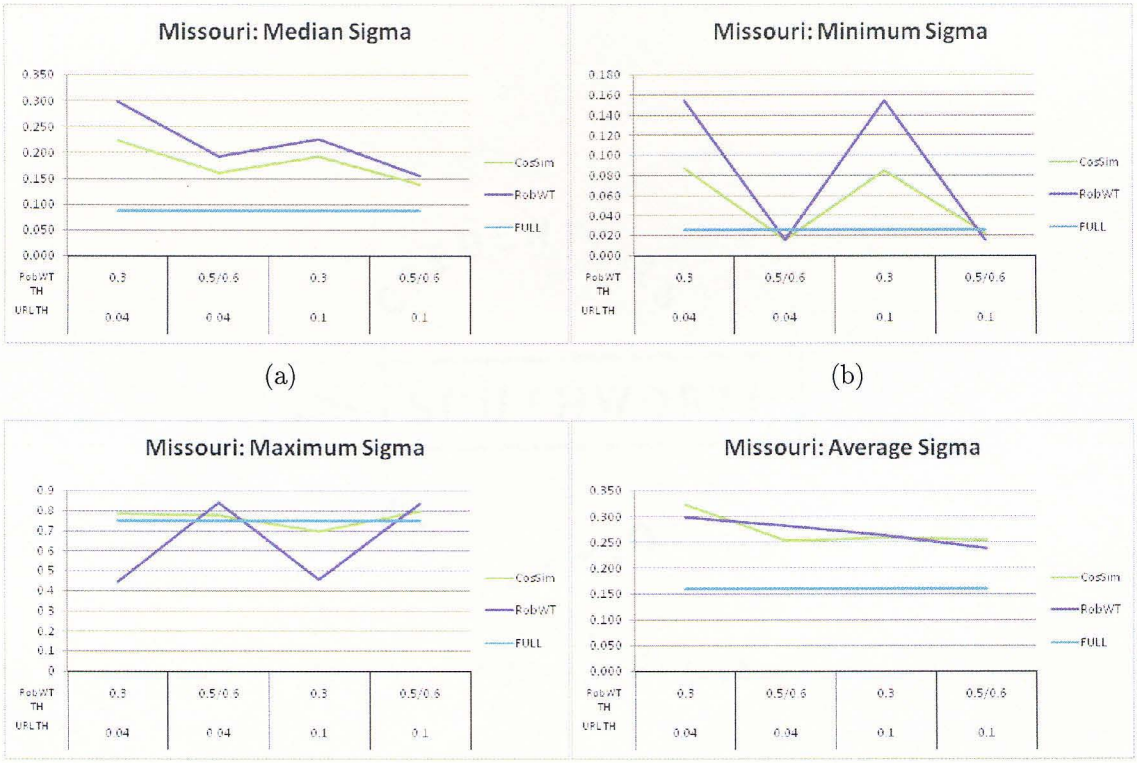
profiles generated using the evolutionary approach showed a comparable density as was seen in Fig.4.17(d).

Fig.4.20 shows the maximum, minimum, average, and median profile variance with each of the methods: Evolution (CosSim, RobWT) and Traditional (Full). The aggregate metrics for the traditional method will stay the same for different configurations, since no threshold values are used. For all aggregates, a low value is desired.

Figs.4.20(a) and (d) show that the median and average variance are always larger using the evolutionary approach (but by only 5-10%). Fig.4.20(b) shows that the minimum variance using the evolutionary approach is larger than the traditional (full) approach when the similarity threshold is low, and is equal or less than the traditional approach when a strict similarity threshold is used. This is because a higher similarity threshold will result in higher quality profiles (i.e. lower variance). The maximum variance in Fig.4.20(c) is similar for the traditional and evolutionary approaches when using a high similarity threshold value, but is less when using Robust Weight Similarity and a low similarity threshold.

The quality of profiles resulting from traditional mining are expected to be of higher quality since *all* log data is mined at once. However, the evolutionary approach has proved that it too can discover profiles that are as good (or better) than using the traditional method. This can be seen in Fig.4.20(b) where the minimum variance for the evolutionary approach was less than that of the traditional approach.

Figure 4.20: Missouri: Evolution vs. Traditional Variance Aggregates



(a) (b)

(c) (d)

4.5 Experiment 2: University of Louisville Library

4.5.1 The Dataset

This experiment will be conduced on a larger and much more recent dataset. The logs were collected for requests done to the University of Louisville's Library main website (library.Louisville.edu).

The logs were collected for five consecutive days: from Wednesday February 27th, 2008 till Sunday March 2nd, 2008. There were a total of 364,409 requests, grouped into 14,888 sessions. The profile evolution tracking was done on a daily basis, so there were five time periods:

- *Wednesday Feb 27th 2008*: 104,794 access requests (28.76%), 4,196 sessions (28.18%)
- *Thursday Feb 28th 2008*: 92,446 access requests (25.37%), 3,657 sessions (24.56%)
- *Friday Feb 29th 2008*: 70,722 access requests (19.41%), 2,919 sessions (19.61%)
- *Saturday March 1st 2008*: 40,834 access requests (11.2%), 1,791 sessions (12.03%)
- *Sunday March 2nd 2008*: 55,613 access requests (15.26%), 2,325 sessions (15.62%)

Fig.4.21 shows the percentage of "bad access" and "good access" requests. Bad access requests include irrelevant and noise requests from search engines requests and requests that resulted in an erroneous status code. Good access requests are the remaining non-error generating requests. Just looking at the figure shows the importance of the preprocessing phase, because all these bad requests (around 85%) would have adversely affected the resulting profiles discovery. Typically graphics requests may amount to 3-4 graphics per page or more (e.g. background picture, top banners, etc), that is why they end up being the majority of requests.

Figure 4.21: U of L Library: Access Requests

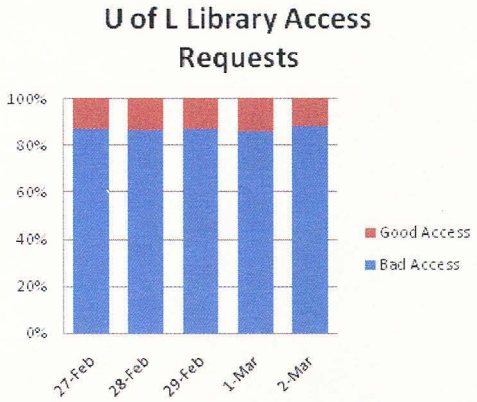


Table 4.5 shows the list of all profiles that resulted from mining this dataset at the last time period (March 2). These profiles are the result of using Robust Weight Similarity with threshold value of 0.6, and URL threshold value of 0.1. Only URLs with weight greater than 0.1 are shown. URLs were sorted in a descending order by the URL significance weight.

4.5.2 Profile Variances

The profiles evolution over time is shown in the Figures 4.22, 4.23, 4.24, and 4.25. Since the number of profiles is large for some configurations (about 50), they were separated in two graphs for the same configuration to avoid clutter: one for the highest cardinality profiles, and the other for the lower cardinality profiles. Separating profiles based on cardinality makes the comparison more accurate, because a low cardinality profile is much more sensitive to changes than a higher cardinality profile.

Fig.4.22 shows the profile variances when using a URL threshold of 0.04 and Binary Cosine Similarity for matching sessions. Figs.4.22(a) and (b) show that the majority of profiles (both high and low cardinality) start with low variance, then increase a little bit until they stabilize, which indicates good quality profiles. However, the

Table 4.5: U of L Library: Final Profiles for RobWT (0.6) and URL TH(0.1)

Profile	Url#	Card	Variance	Content
0	2	667	0.0229	/, /promo/bookplates.swf,
1	78	1747	0.291953	/kornhauser,
2	5	157	0.088875	/research/sub/dbsA.html, /research/sub/dbsDE.html, /research/sub/dbsP.html,
3	5	724	0.100414	/, /research/sub/dbsA.html, /promo/bookplates.swf,
4	7	102	0.056861	/research/sub/dbsA.html, /research/sub/dbsJKL.html, /, /promo/bookplates.swf, /research/sub/dbsDE.html,
5	18	61	0.186887	/research/sub/dbsA.html, /top/subjects.html, /, /promo/bookplates.swf, /research/sub/dbsDE.html, /research/sub/dbsP.html, /research/sub/dbsJKL.html, /research/business/index.html, /top/tools.html, /research/panafrican/index.html, /research/health/index.html, /research/sub/dbsNO.html,
6	6	45	0.059365	/research/sub/dbsA.html, /research/sub/dbsM.html, /, /promo/bookplates.swf,
7	1	386	0.005911	/music,
8	2	100	0.00968	/music/listenonline.html, /music,
9	6	77	0.056598	/, /music, /promo/bookplates.swf, /music/listenonline.html,
10	2	264	0.110867	/government/news/otherlinks/otherlinks.html, /,
11	4	56	0.112673	/ekstrom, /research/sub/dbsA.html,
12	3	65	0.055307	/art, /promo/bookplates.swf,
13	1	159	0.007119	/forms/z.htm,
14	1	154	0.004239	/ekstrom/special/moi/moi-people01.html,
15	9	96	0.224964	/top/subjects.html, /, /research/communication/index.html, /research/anthropology/index.html, /promo/bookplates.swf, /research/general/index.html,
16	2	64	0.139356	/government/periodicals/periodall.html,
17	3	1166	0.071876	/, /promo/bookplates.swf,
18	5	83	0.02923	/research/sub/dbsA.html, /research/sub/dbsP.html, /, /promo/bookplates.swf,
19	5	36	0.058454	/research/sub/dbsJKL.html, /research/sub/dbsA.html, /, /ekstrom,
20	8	78	0.094318	/research/sub/dbsA.html, /research/sub/dbsDE.html, /, /promo/bookplates.swf, /research/sub/dbsJKL.html,
21	8	41	0.108382	/research/sub/dbsWXYZ.html, /research/sub/dbsA.html, /, /promo/bookplates.swf, /top/subjects.html,
22	10	49	0.108915	/research/sub/dbsA.html, /research/sub/dbsS.html, /, /promo/bookplates.swf, /research/sub/dbsWXYZ.html, /top/subjects.html, /research/sub/dbsDE.html,
23	11	90	0.252119	/research/sub/dbsA.html, /research/sub/dbsNO.html, /, /research/sub/dbsB.html, /top/subjects.html,
24	3	99	0.052191	/ill, /,
25	2	48	0.142716	/government/goodsources/factbook.html,
26	2	49	0.03262	/art, /,
27	8	45	0.180277	/ekstrom, /, /ekstrom/hours/index.html, /job, /services/index.html,
28	4	105	0.322663	/government/news/otherlinks/otherlinks.html,
29	2	450	0.0233	/promo/bookplates.swf, /,
30	6	62	0.094646	/research/sub/dbsA.html, /research/sub/dbsS.html, /research/sub/dbsWXYZ.html, /research/sub/dbsC.html,
31	4	445	0.120767	/, /research/sub/dbsA.html,
32	5	39	0.054863	/research/sub/dbsA.html, /research/sub/dbsDE.html, /,
33	4	41	0.043474	/research/sub/dbsA.html, /research/sub/dbsNO.html, /, /research/sub/dbsDE.html,
34	4	50	0.0783	/research/sub/dbsA.html, /research/sub/dbsP.html, /research/sub/dbsJKL.html, /top/subjects.html,
35	1	58	0.024124	/ill,
36	1	35	0.011143	/art,
37	6	33	0.129325	/dlls, /dlls/database, /, /dlls/guide, /dlls/forms,
38	3	549	0.080418	/, /promo/bookplates.swf,
39	7	124	0.140677	/research/sub/dbsA.html, /, /research/sub/dbsP.html, /research/sub/dbsJKL.html, /research/sub/dbsDE.html, /research/sub/dbsNO.html,
40	8	41	0.140021	/ekstrom, /, /promo/bookplates.swf, /ekstrom/hours/index.html, /kornhauser,
41	1	41	0.01061	/kornhauser/forms/webform.html,
42	2	705	0.085	/, /promo/bookplates.swf,
43	4	48	0.2716	/ekstrom, /top/subjects.html, /ekstrom/hours/index.html,
44	9	182	0.34	/research/sub/dbsA.html, /top/subjects.html, /research/sub/dbsDE.html, /research/business/index.html, /, /research/sub/dbsP.html, /research/sub/dbsJKL.html, /promo/bookplates.swf,
45	2	29	0.0873	/government/states/kentucky/kylit/berry.html, /government/states/kentucky/kylit/berryadd.html,
46	5	30	0.4754	/government/news/otherlinks/otherlinks.html, /government/subjects/health/daterape.html,

top three profiles (17, 18, and 26) have the highest cardinality and a relatively high variance, and they don't seem to improve over time, i.e. their variance stays almost the same during evolution.

When using a stricter similarity threshold in Figs.4.22(c) and (d), the quality of profiles seems to improve. The majority of profiles start with and keep a stable low variance over time, and the high cardinality profiles even seem to improve their variances, e.g. profile number 1. Therefore, an initial conclusion from these results is that a stricter similarity threshold resulted in better quality profiles, since only close sessions are used to update profiles.

Fig.4.23 shows the profiles evolution for a higher URL threshold (0.1) and using Binary Cosine Similarity. Figs.4.23(a) and (b) shows that all profiles are of high quality with variance less than 0.5. However, some of them have a tendency to increase their variance. Figs.4.23(c) and (d) (with strict matching) show more stable variance values with the majority under 0.3, with the exception of the profile with highest cardinality (profile 1) which starts with high variance and later improves its variance. The results shown in this figure point to a more desirable behavior than the ones in Fig.4.22. Hence, a higher URL threshold caused more stability and better quality of profiles.

For the Robust Weight Similarity, Fig.4.24 shows the variance evolution when using 0.04 as the URL weight threshold value. For 0.3 similarity threshold, fewer profiles are developed and their profile variances are shown in Fig.4.24(a), which shows that profiles starting with low cardinality have increased their variances slightly, while profiles with higher cardinality started with high variance and then decreased their variances, with most profiles finally stabilizing around an average variance of 0.5.

Figure 4.22: U of L Library: Evolution of Profile Variances (CosSim , URL TH(0.04))

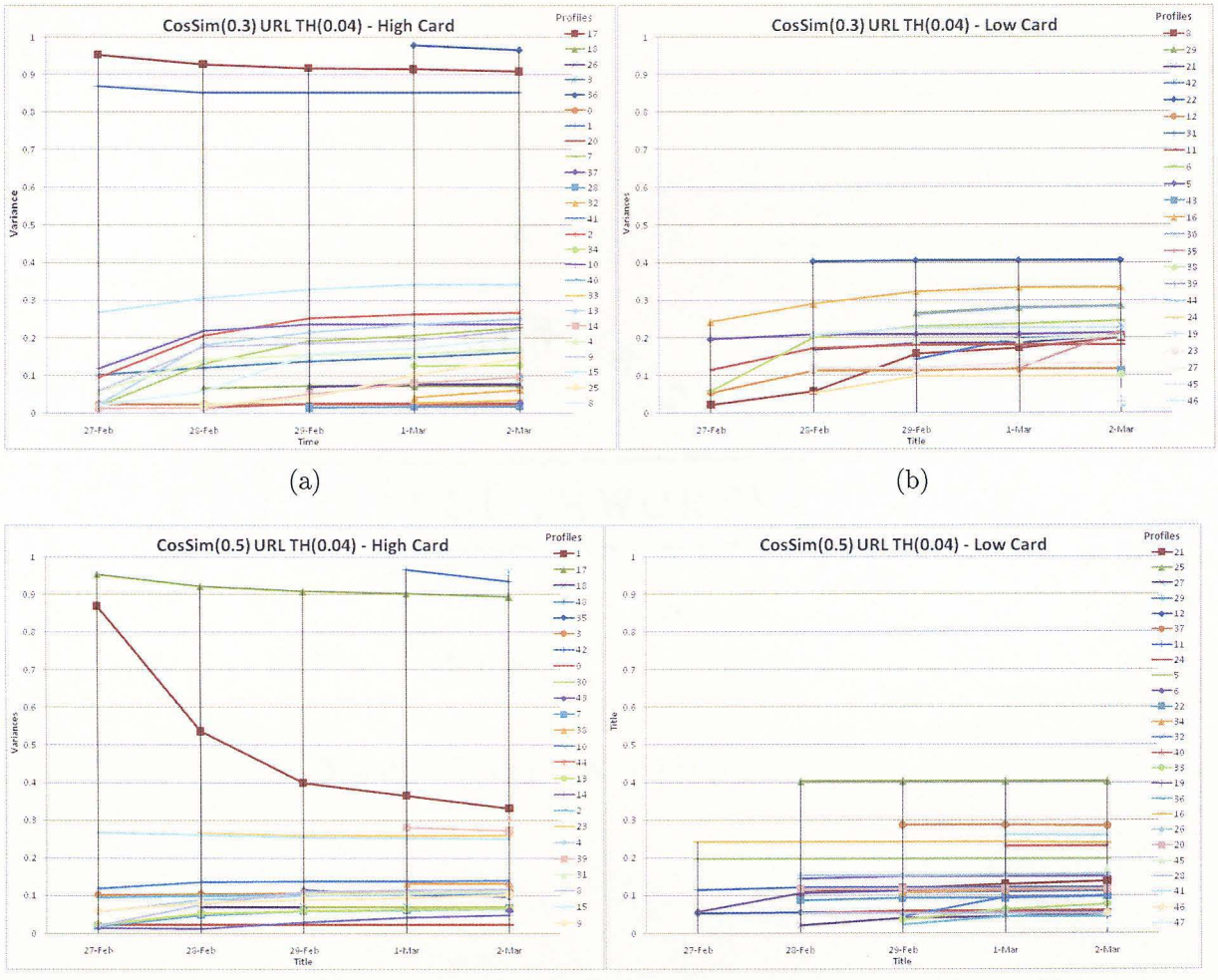
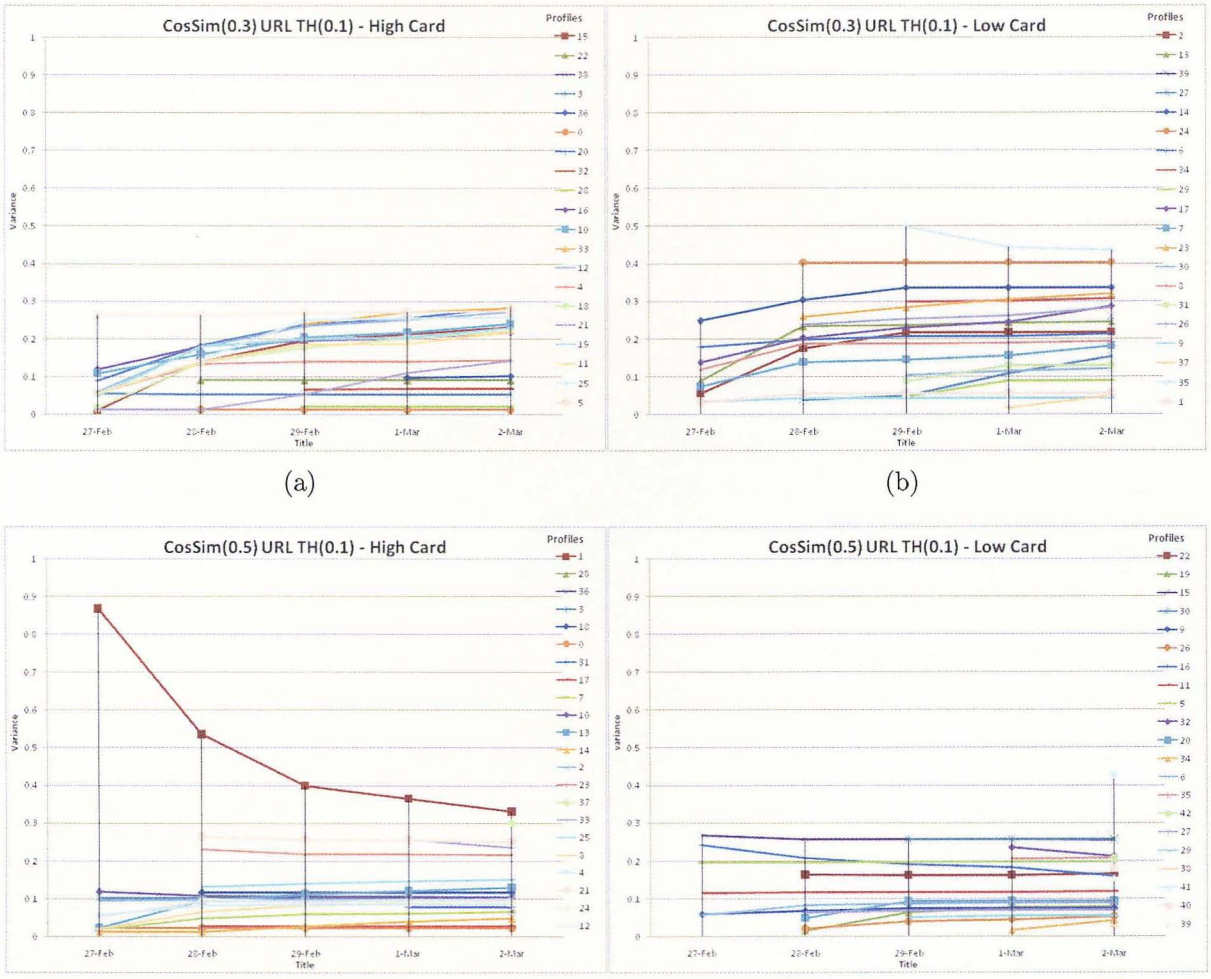


Figure 4.23: U of L Library: Evolution of Profile Variances (CosSim, URL TH(0.1))



Figs.4.24(b) and (c) show that more profiles were discovered with a higher matching threshold. The majority of profiles have stable variance changes, except the highest cardinality profiles which start with high variance that later start decreasing slightly. As in Fig.4.22, we can conclude that a stricter matching threshold has resulted in higher quality and more detailed profiles. Comparing the results from this configuration to Figs.4.22(c) and (d), we can see that both seem to show similar behavior. Thus, using the two methods of similarity (CosSim and RobWT) with a more restricted threshold value can be expected to cause the majority of profiles to have low and stable variance evolution over time.

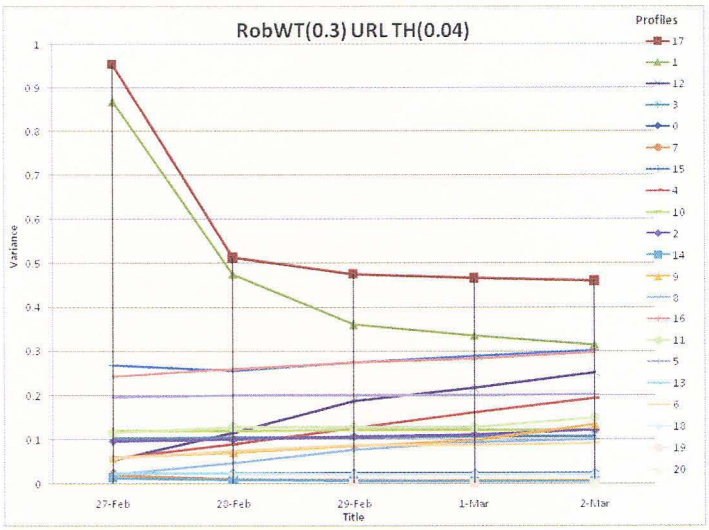
Fig.4.25 shows the variances when using a more restrictive URL threshold value of 0.1. Fig.4.25(a) shows a similar behavior to Fig.4.24(a), where a low number of profiles were discovered with the majority of profiles having low variances, that increased slightly with the exception of the highest cardinality profile, for which the variance decreased from a high value to an average value.

The last configurations in Figs.4.25(b) and (c) show the best quality of profiles, because most profiles began with a small variance and they kept improving , in contrast with the other configurations where the variances either remained the same or increased. Even the high cardinality profiles which started with high variance, drastically decreased in their variance with time to a low value.

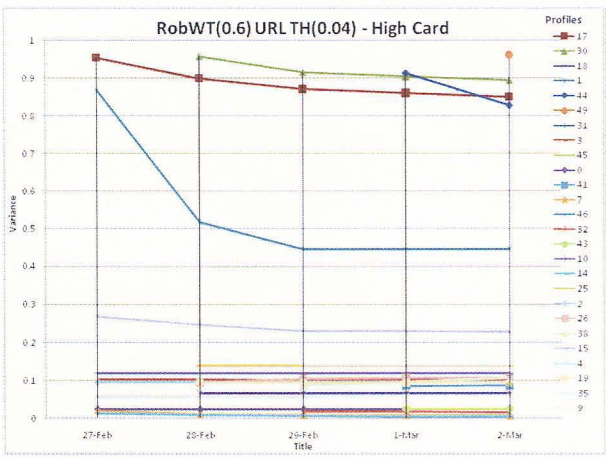
An overall conclusion can be drawn that more restrictive threshold values combined with the use of the profile-sensitive Robust Weight Similarity tend to result in very high quality profiles over time.

To study the overall evolution behavior profile quality, Fig.4.26 shows the minimum, maximum, median, and average variance of the final profiles at the end of

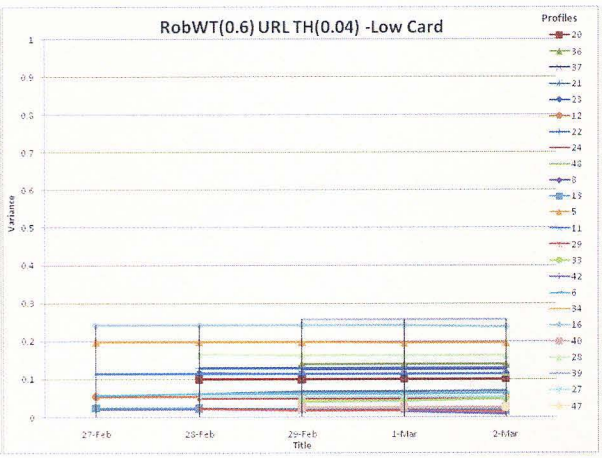
Figure 4.24: U of L Library: Evolution of Profile Variances (RobWT , URL TH(0.04))



(a)

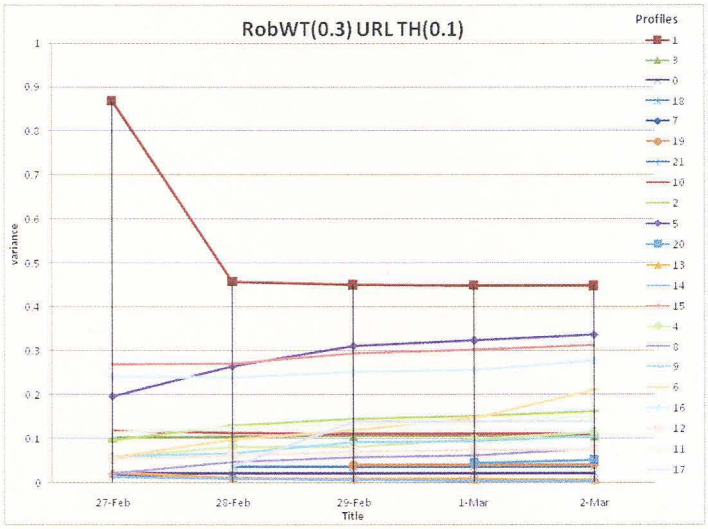


(b)

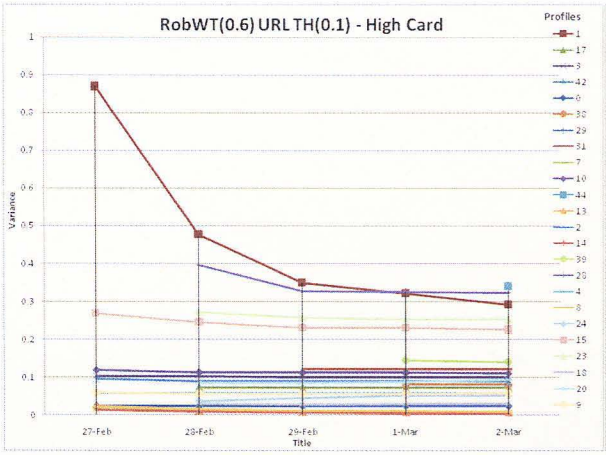


(c)

Figure 4.25: U of L Library: Evolution of Profile Variances (RobWT , URL TH(0.1))



(a)



(b)

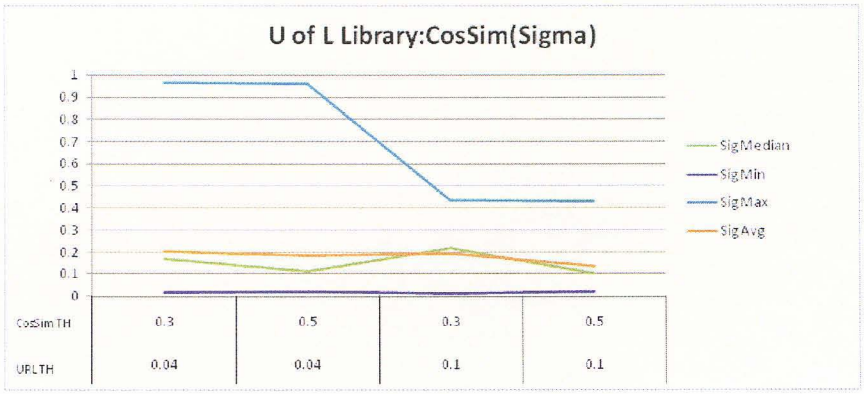
(c)

evolution. As was noticed in Fig.4.5 for the smaller Missouri dataset, the minimum, median, and average variance remained stable and within the same range for all different configurations. The median and average variances indicate that the overall quality of profiles is good, since their values are low (<0.2). However, this result should be taken in context with the final number of profiles which will be discussed next.

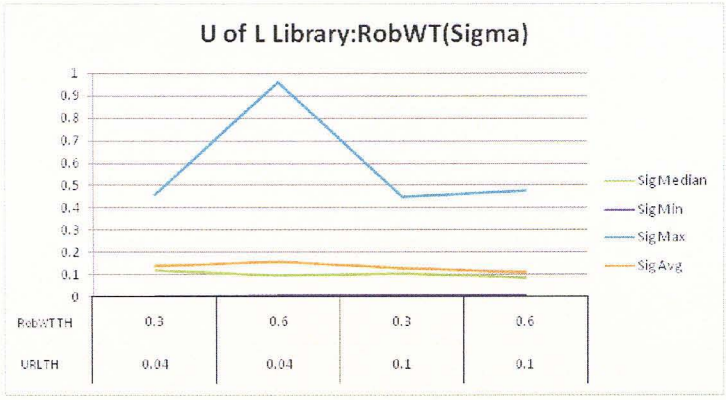
Fig.4.26(a) shows that moving the URL weight threshold value from 0.04 to a more restrictive value of 0.1 caused the maximum variance to drop drastically. On the other hand, the value of the cosine similarity threshold did not seem to have any effect. The decrease in variance can be explained because the binary cosine similarity is computed based on the number of common URLs between a profile and a session. So a larger number of URLs (due to a lower URL threshold) will help increase this similarity and hence increase the chance of sessions passing the thresholding test. Since URL weight threshold is applied in both post-processing and updating phases, all profiles would have either more or less URLs. In case of using a low value for URL weight threshold, the number of URLs in the profile is more than when using a 0.1 threshold value. Therefore more sessions will pass the thresholding test, and lower quality profiles are thus generated. This is why the maximum variance value is more sensitive to the URL weight threshold.

Fig.4.26(b) shows an opposite behavior, where the similarity threshold has more effect on the value of the maximum variance. In contrast to the Binary Cosine Similarity, the Robust Weight Similarity takes into account the profile variance, not just the number of URLs in the profile. Hence, if the profile has low variance, it becomes harder for new sessions to match the profile. For this reason, changing the number of URLs in the profile by changing the URL threshold value will not have the same affect as it had when using the Binary Cosine Similarity.

Figure 4.26: U of L Library: Profiles Sigma Aggregates

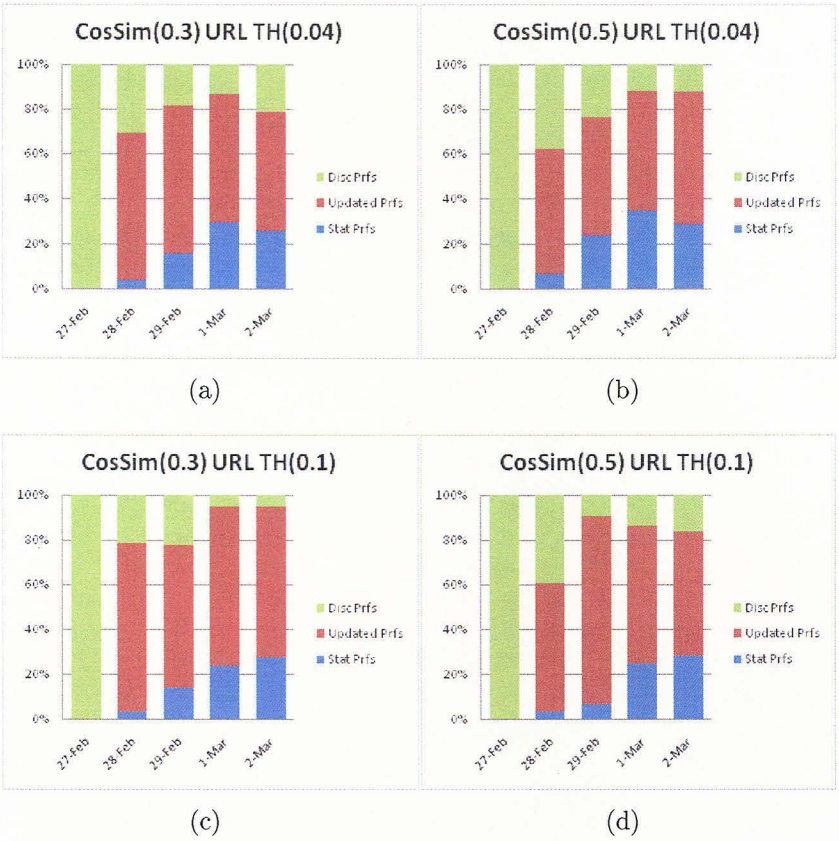


(a)



(b)

Figure 4.27: U of L Library: Profile Counts (Binary Cosine Similarity)



(a) (b)

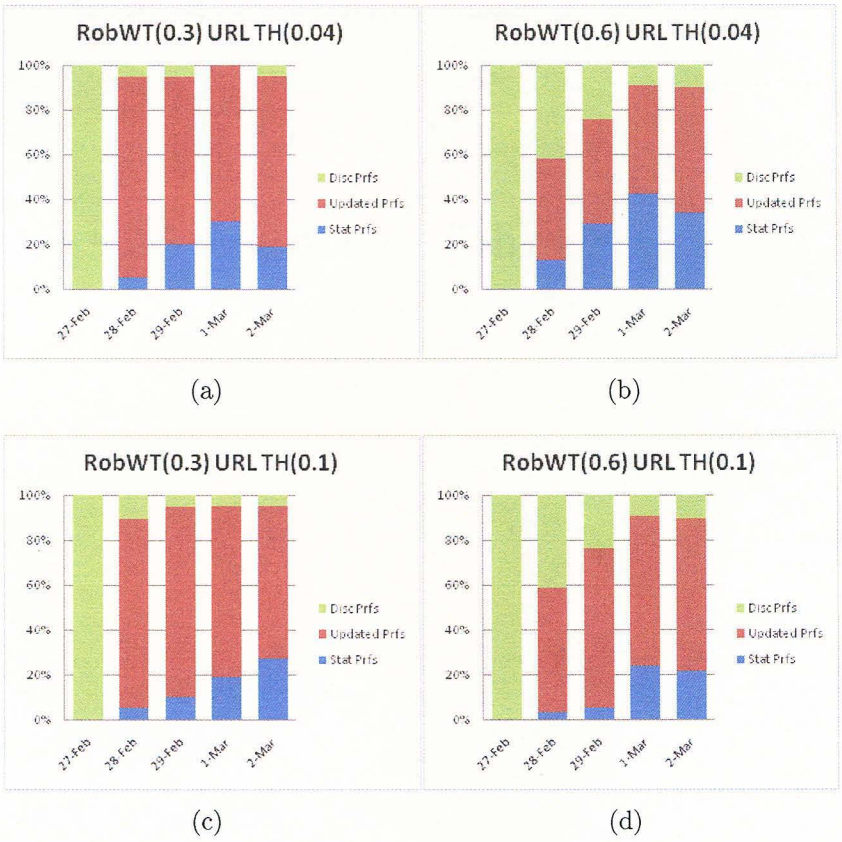
(c) (d)

4.5.3 Profile Evolution

The percentage of each type of profiles discovered/updated for each time period when using the Binary Cosine Similarity is shown in Fig.4.27. The percentages of profiles seem to be similar using the different configurations, with the exception of Fig.4.27(c), where more profiles got updated on Feb 28th (Day 2) , and fewer distinct profiles where discovered on March 2nd (Day 5). In all four configurations, the majority of profiles got updated, while a few profiles were discovered.

Figure 4.28 shows the profile percentages when using the Robust Weight Similarity for matching. Figs.4.28(a) and (c) show similar behaviors, where the vast majority of profiles got updated during all time periods. This can be explained for the low

Figure 4.28: U of L Library: Profile Counts (Robust Weight Similarity)



(a) (b)

(c) (d)

similarity threshold value of 0.3, which allows more sessions to match existing profiles and thus update them.

Fig.4.28(b) and (d) show more restricted profile updates, with more new profiles being discovered or old ones remaining the same. This is because the higher value of similarity threshold decreases the chance that a new session would match an existing profile, and hence would either create a new profile if enough such sessions agree on a usage pattern, or would be discarded otherwise.

4.5.4 Matching vs. Distinct sessions

For a fixed configuration, the number of matching and distinct sessions may shed

a light on which time periods have more usage behavior changes. Fig.4.29 shows the percentages of matching and distinct session for each time period when using the binary cosine similarity. All configurations show that at each time period, the percentage of matching sessions was between 15% and 25%. These results can give a better understanding about the real percentages of profiles and their types as discussed in the previous section, because Fig.4.27 showed that the majority of profiles where updated, which seems to contradict what this figure suggests since the majority of sessions are counted as distinct, so it seems that the majority of profiles should be newly discovered instead of updated. This contradiction is due to the fact that more distinct sessions do not necessary imply more new profiles to be discovered. For example all the distinct sessions might be very close to each other, and thus generate only one distinct profile.

Figure 4.30 shows the number of matching and distinct sessions when using the Robust Weight Similarity. Fig.4.30(a) and (c) show similar behavior where the vast majority of sessions got matched. This is due to the loose similarity threshold value of 0.3, which is compatible with the findings in Fig4.28(a) and (c) where most profiles got updated.

Fig.4.30(b) and (d) show a completely opposite behavior to Fig.4.30(a) and (c) where the majority of sessions are treated as distinct. This is reflected in Fig.4.28(a) and (c) where some profiles were distinct. As discussed above, the large number of distinct sessions does not necessarily mean that a large number of new profiles will be discovered.

The overall trend of matching sessions is shown in Fig.4.31, which shows that for the binary cosine similarity, the percentage of matching sessions is similar for all

Figure 4.29: U of L Library: Matching vs. Distinct Sessions (Binary Cosine Similarity)

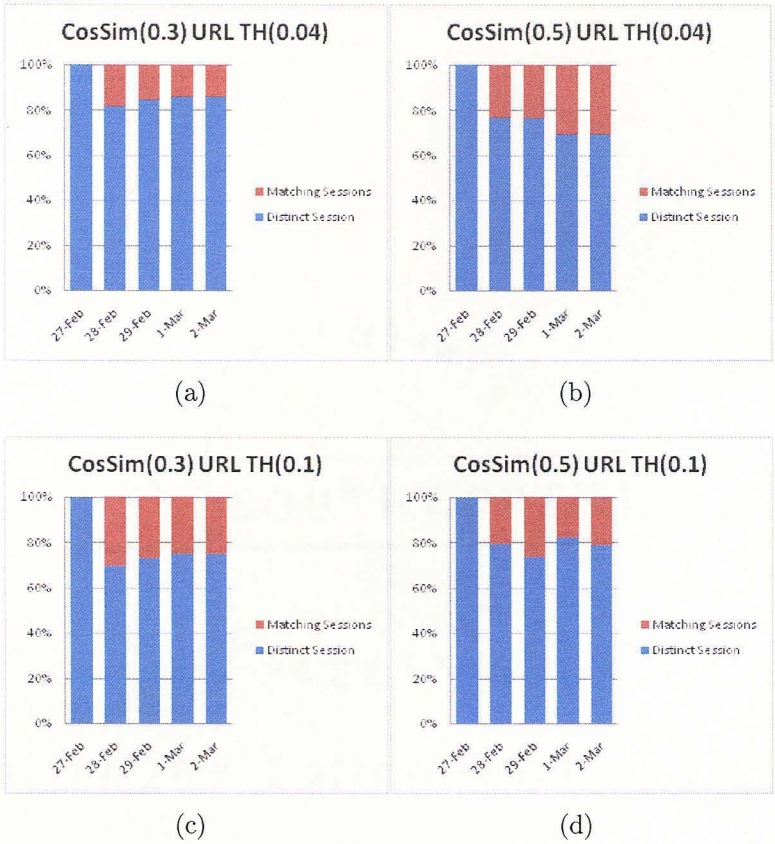


Figure 4.30: U of L Library: Matching vs. Distinct Sessions (Robust Weight Similarity)

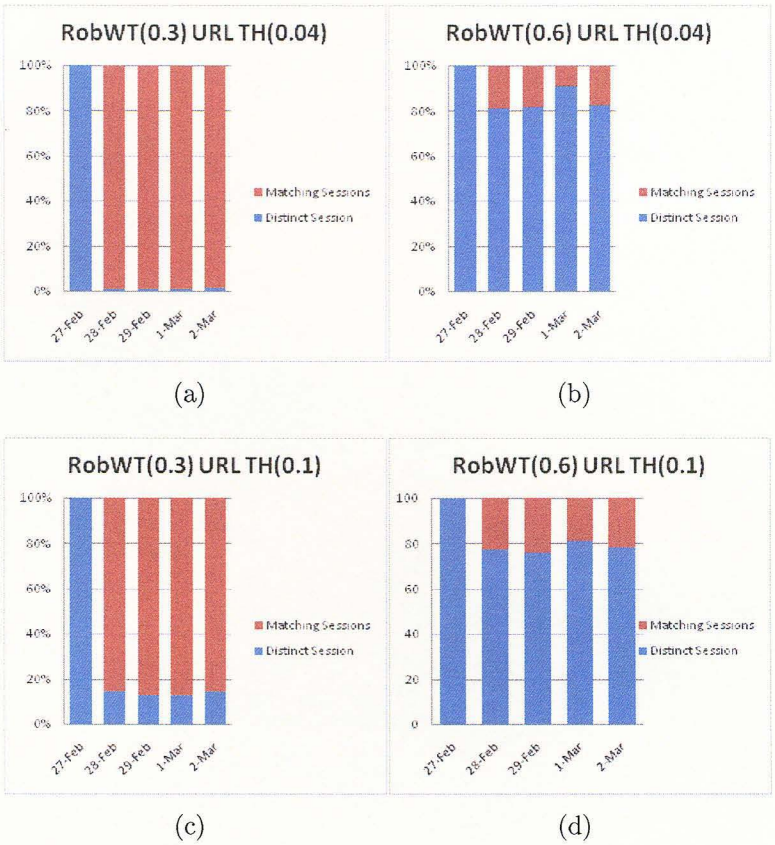
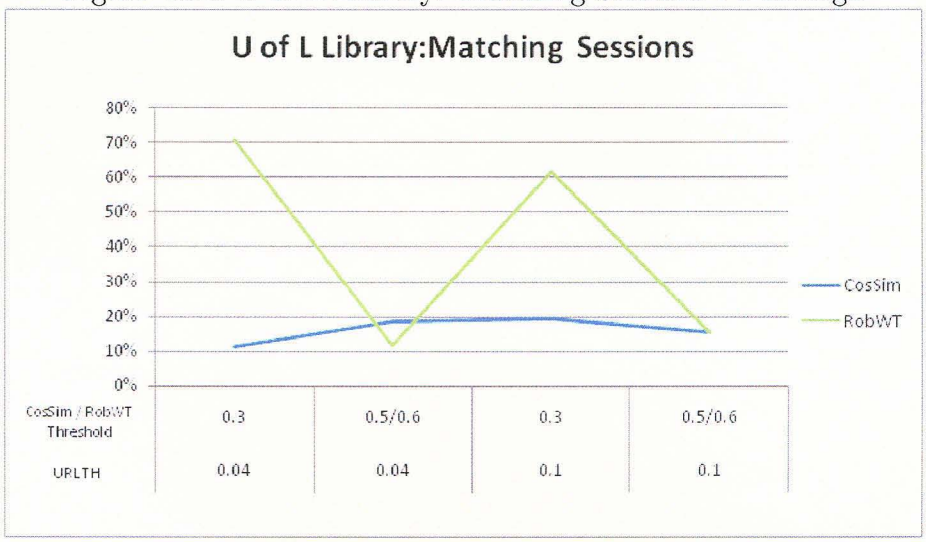


Figure 4.31: U of L Library: Matching Sessions Percentage



configurations. However, for the robust weight similarity, the percentage gets much higher when the matching threshold is lower.

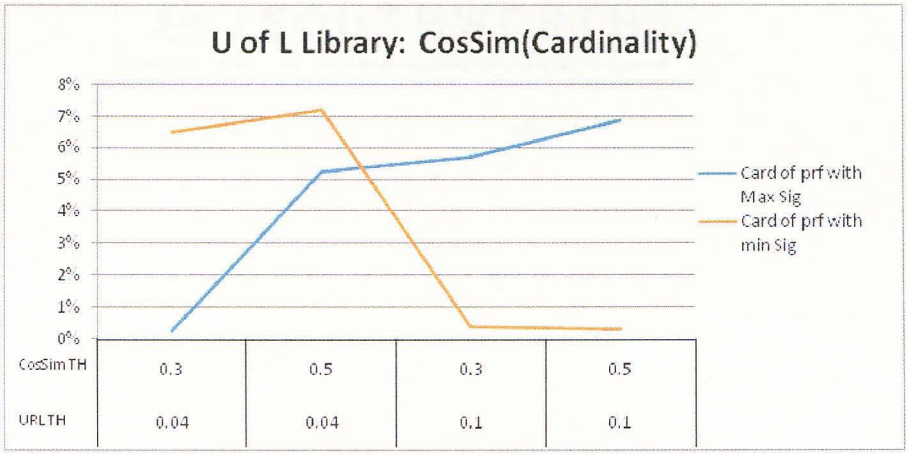
4.5.5 Profile Cardinality

The profile cardinality represents how popular and important a profile is. In Fig.4.32, the cardinality of the profiles with maximum and minimum variance is plotted for each configuration.

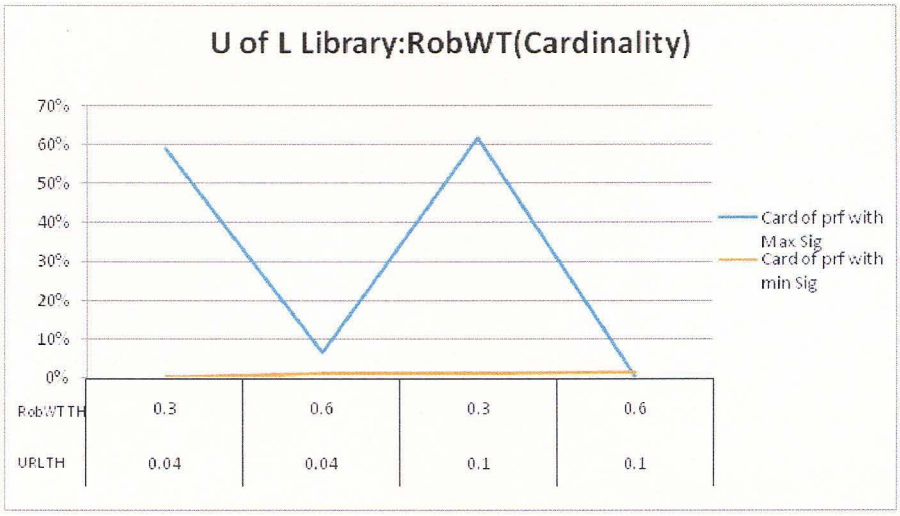
Fig.4.32(a) presents the results when using the binary cosine similarity, and it shows that the lower URL weight threshold caused the highest quality profile (i.e. with minimum variance) to have a high cardinality, while a higher URL threshold value caused the high quality profiles to have fewer sessions. However, when it comes to the lowest-quality profile (i.e. the one with high variance), the similarity threshold value was the main factor affecting the cardinality, since a higher threshold causes the profile with maximum variance to acquire more sessions.

Fig.4.32(b) shows the results when using the robust weight similarity. The cardinality of the high quality profile (with low variance) remains the same for all different

Figure 4.32: U of L Library: Cardinality of Max and Min Sigma



(a)

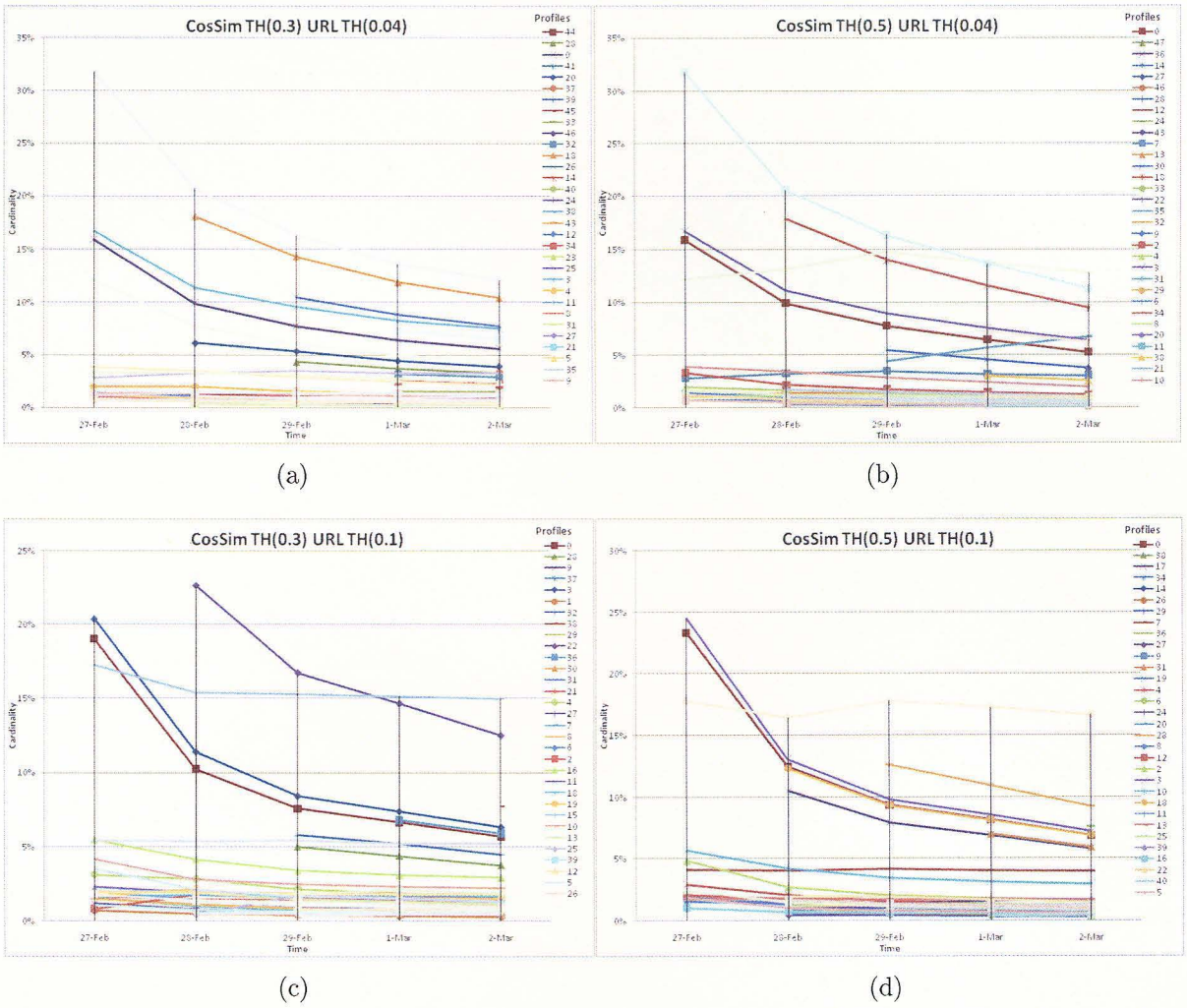


(b)

configurations. However, the cardinality of the profile with maximum variance is sensitive only to the similarity threshold value, where a high threshold value causes the cardinality to decrease, while a lower threshold value causes higher cardinality.

For a more detailed view of the profile quality, the cardinalities of each profile at each time period are plotted. Fig.4.33 shows the cardinality percentage of each profile at each time period when the Binary Cosine Similarity. The cardinality of the

Figure 4.33: U of L Library: Evolution of Profile Cardinality Percentages (CosSim)



(a) (b)

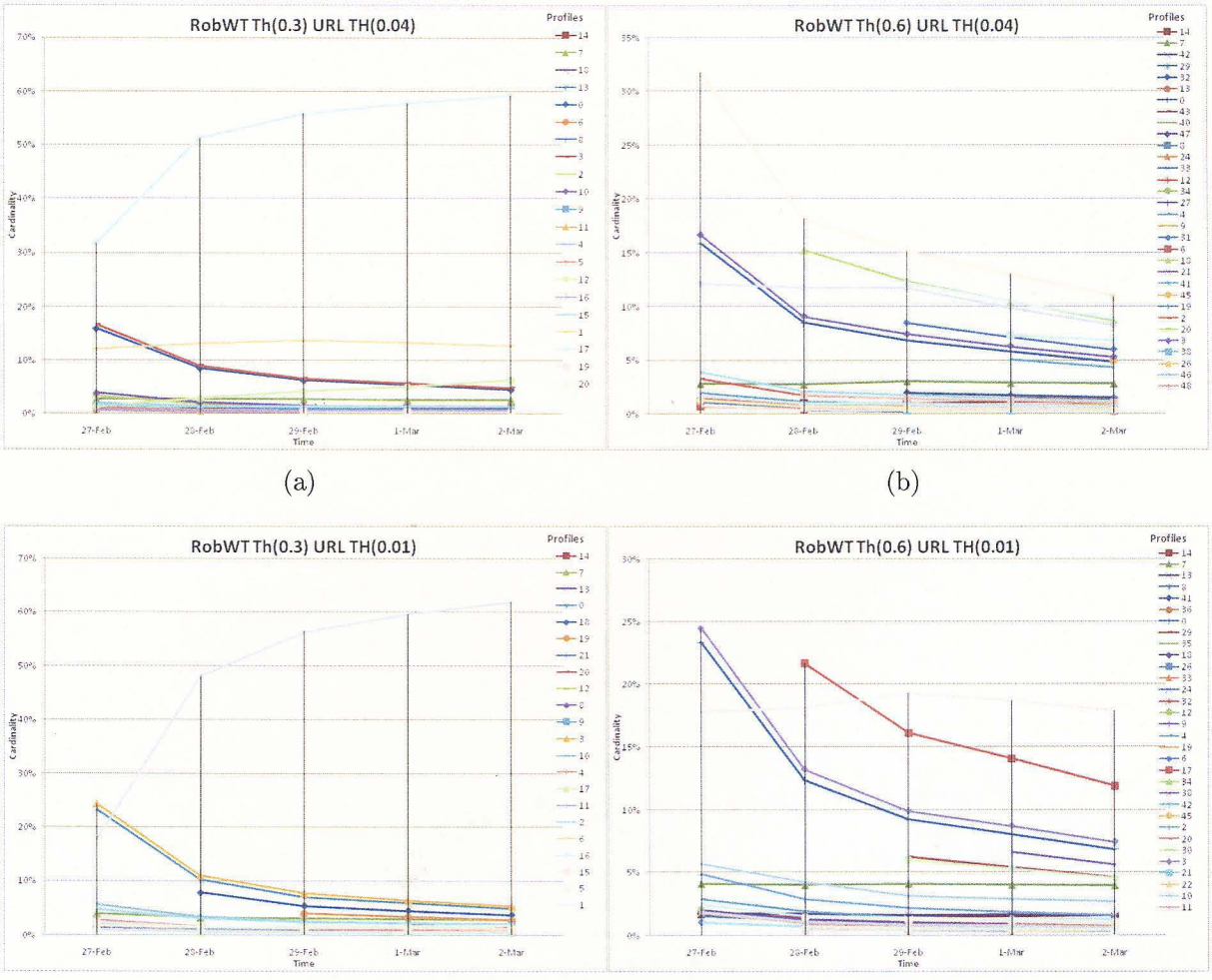
(c) (d)

profile at each time period represents all sessions that matched this profile up to (and including) that time period. The cardinality is always normalized by the sum of all profiles cardinalities. The profiles are ordered (in a decreasing order) based on their variance.

For all configurations, the majority of profiles are slightly decreasing in their cardinality over time. This means that there is a high competition between profiles over new sessions.

Fig.4.34 shows the profile cardinalities over time when using the robust weight

Figure 4.34: U of L Library: Evolution of Profile Cardinality Percentages (RobWT)



(a) (b)

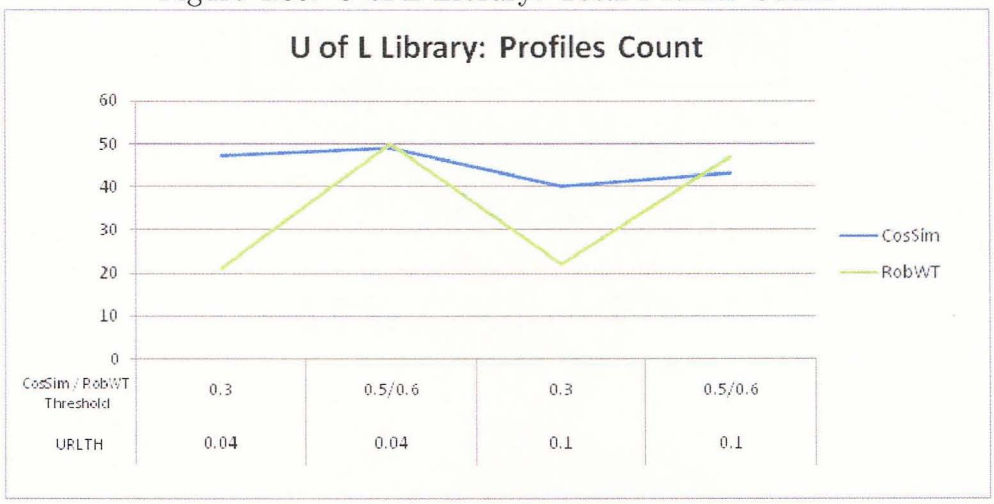
(c) (d)

similarity. As seen in Fig.4.33, the majority of profiles for all configurations slightly decrease their cardinality over time, with the exceptions of two profiles: one in Fig.4.34(a) and one in Fig.4.34(c). Despite the increasing in their cardinality, these two profiles have high variance, which means that they are of low quality; most likely, they are the "default" clusters where all non-matching sessions are assigned.

4.5.6 Profiles Count

The number of profiles generated from the usage behavior is shown in Fig.4.35.

Figure 4.35: U of L Library: Total Profiles Count



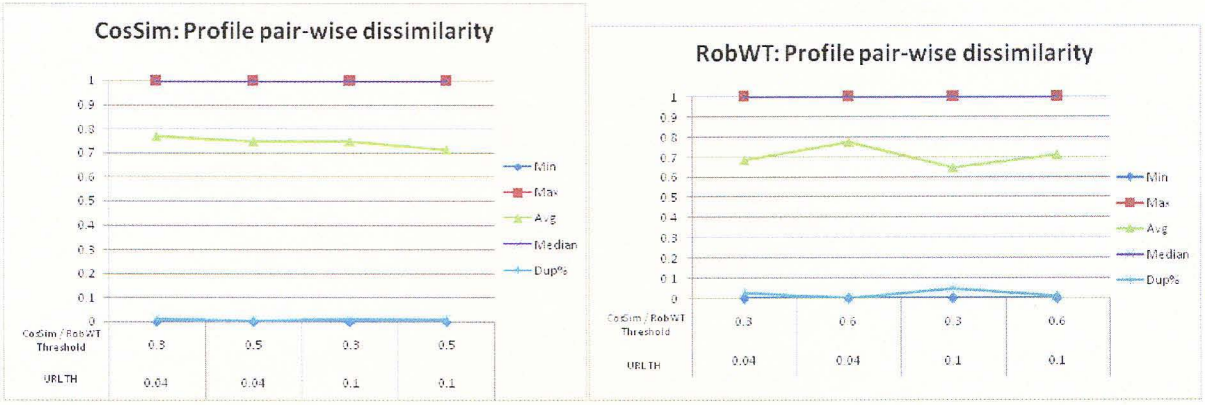
Since HUNC is a hierarchical algorithm, a high number of profiles indicates more detailed profiles, which is desirable. Generally, detailed profiles are of good quality, while a general profile can be considered of low quality, and should be split into multiple detailed profiles. However, the larger number of profiles typically causes low cardinality, since the sessions are distributed over more profiles. Thus, these profiles become more sensitive to any updates and might change their behavior more often.

The number of profiles, when using the binary cosine similarity, is mostly the same for different configurations, whereas using the robust weight threshold generates a lower number of profiles for the lower (0.3) threshold value, and a larger number of profiles for the higher (0.6) threshold value. A more detailed analysis of the profile types was done in Section 4.5.3.

4.5.7 Profile Pair-wise Similarity

A powerful evaluation metric is to find how similar the profiles that result from the pattern discovery are to each other. If profiles are too similar to each other, then this would indicate that the updating algorithm was not able to match the new logs

Figure 4.36: U of L Library: Profile Pair-wise Similarity Aggregates



(a) (b)

with the existing profiles accurately.

A difference matrix consisting of the pair-wise similarities between every two profiles at the end of evolution was calculated. The maximum, minimum, average, and median pair-wise similarities are plotted on Fig.4.36, as well as the percentage of duplicate profiles. Duplicate profiles are those whose difference is less than 0.01. This translates to profiles who are at least 90% similar to each other (using the binary cosine similarity).

Both Figs.4.36(a) and (b) show similar trends, where the maximum and median dissimilarity is one (which means the majority of profiles are completely different), while the average difference is around 0.7 which can be translated to profiles which are only 16% similar. The minimum profile dissimilarity is close to zero, which means that these two profiles are too similar to each other. The percentage of duplicate profiles is almost zero in all cases (it is zero in some configurations), which means that the total number of duplicate profiles is extremely low, which supports the objective of this thesis of developing high quality, distinct, and evolving profiles through different time periods.

4.5.8 Profile Density

Fig.4.37 shows the profile densities over time when using the Binary Cosine Similarity. The profiles were ordered based on their variance (in decreasing order).

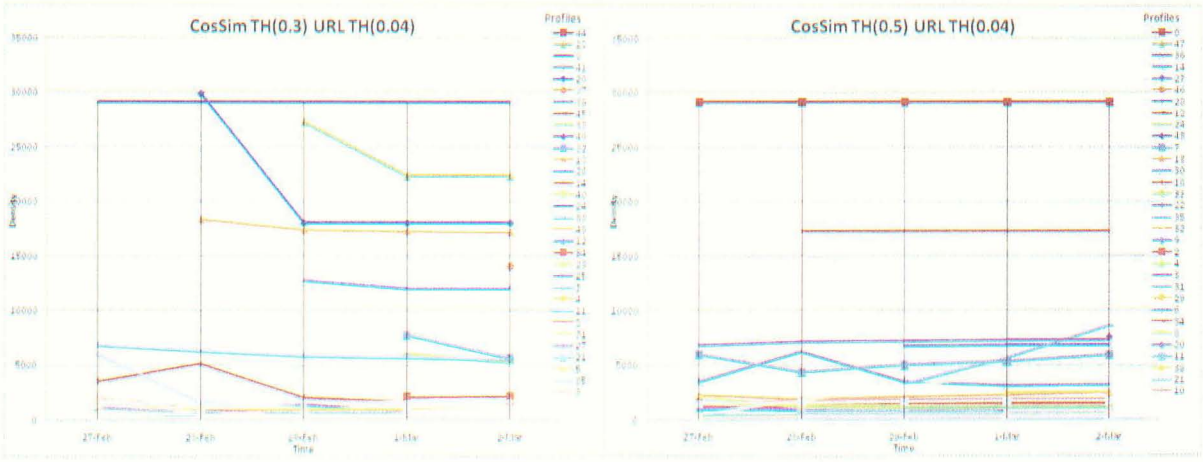
Most profiles maintain a stable density over time for all configurations. Some of them decrease their density slightly as in Fig.4.37(a), which means that the variance is increasing (since the cardinality does not decrease over time), and this increase in variance is due to the profile being updated by new sessions which are not very similar to existing ones in the profile. This happens because of the low similarity threshold value used (0.3).

Fig.4.38 shows the profile densities when using the Robust Weight Similarity. All configurations show better results than the results shown in Fig.4.37 which uses the Binary Cosine Similarity instead. In Fig.4.38 more profiles are improving their quality (increase their density) over time. This supports conclusions reached in figures 4.24 and 4.25, where the variances are decreasing over time, and profiles are becoming more compact (hence of higher quality). This consistent increase in densities may be the ultimate judge that matching using the Robust Weights (which depend on accurate variance estimates) yields the best quality profiles compared to Cosine Similarity matching.

4.5.9 Evolution vs. Traditional (Full run)

The last criterion is the approach of pattern discovery, which is either *evolutionary* -discussed in this thesis- where the pattern discovery is done through different time periods, or *traditional* where the log data from all time periods is used in one shot to discover usage patterns.

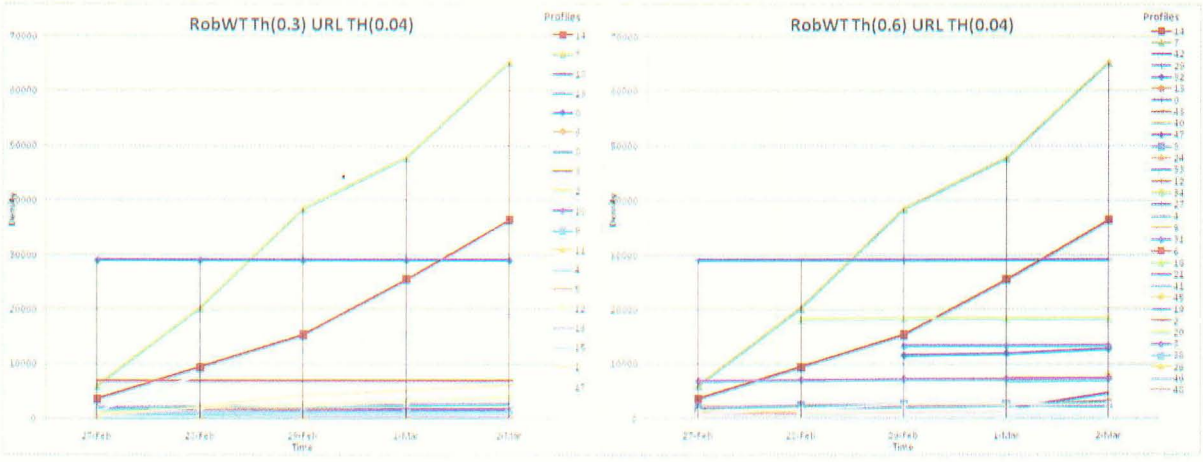
Figure 4.37: U of L Library: Evolution of Profile Densities (CosSim)



(a) (b)

(c) (d)

Figure 4.38: U of L Library: Evolution of Profile Densities (RobWT)



(a) (b)

(c) (d)

Figure 4.39: U of L Library: Traditional Profile Variances

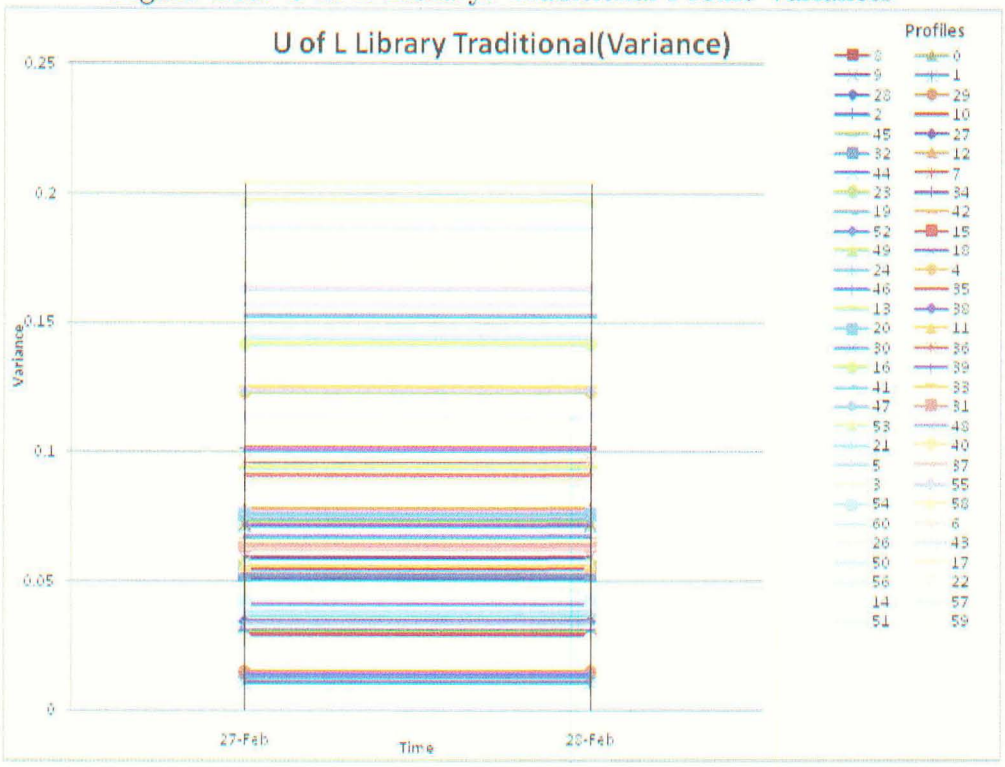


Fig.4.39 shows the profile variances when using the traditional discovery approach. Profiles are ordered in a decreasing order based on their cardinality percentage. All profiles are of good quality since their variance is low, where the highest variance is only about 0.2.

Fig.4.40 shows the densities of the profiles discovered when using the traditional discovery approach. The density represents the ration between the profile cardinality and its variance. Profiles are order in a decreasing order based on their cardinality percentage. All profiles are of high quality since the density is high. Some of the profiles generated using the evolutionary approach showed a comparable or higher density as was seen in Fig.4.38.

Figure 4.40: U of L Library: Traditional Profile Density

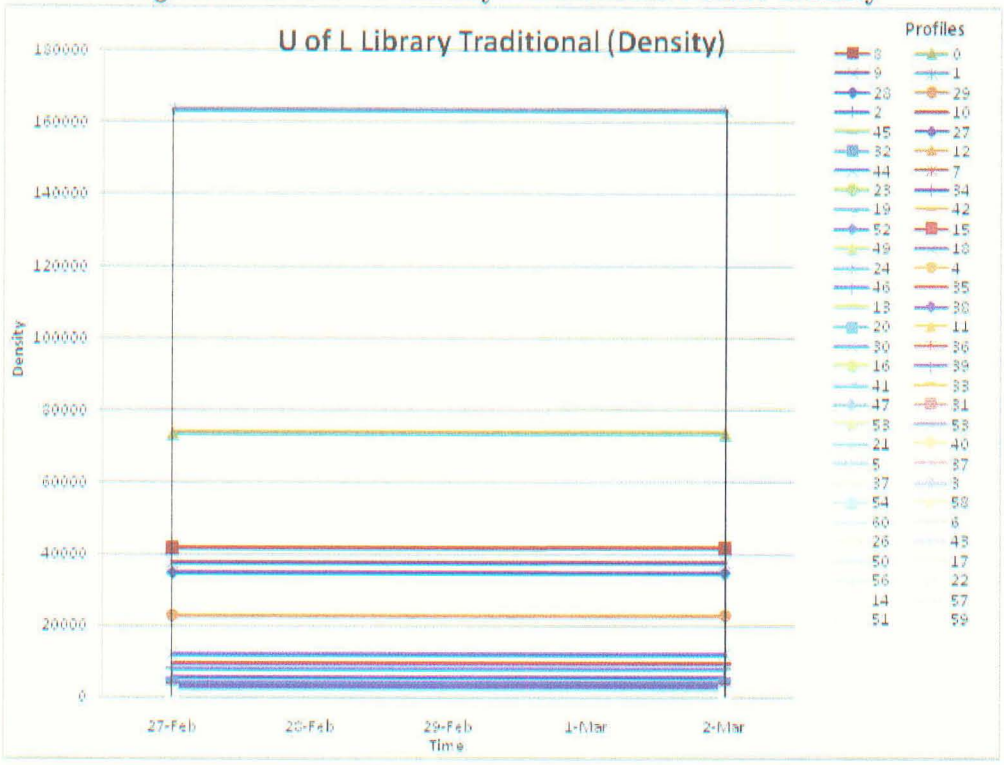
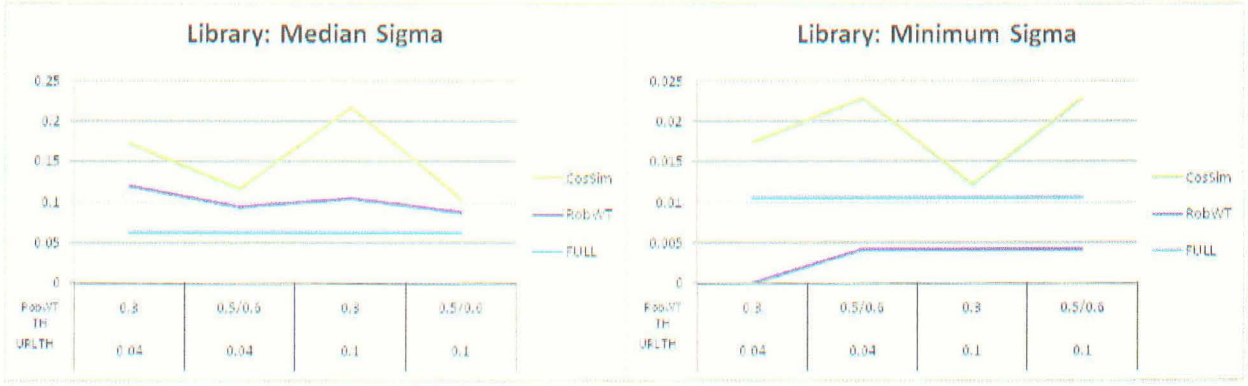


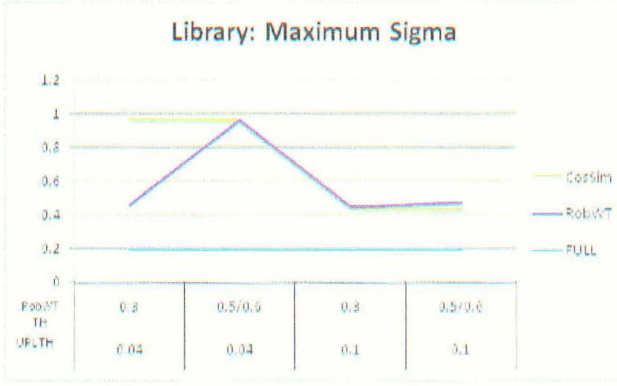
Fig.4.41 compares the aggregate metrics for both discovery approaches (traditional and evolution). The median, maximum, and average variances are less when using the traditional approach than using the evolutionary approach. Fig.4.41(b) shows that using the Robust Weight Similarity in the evolution mode has resulted in the minimum profile variance for all configurations compared to using the Evolution mode's Cosine Similarity and compared to the traditional mode. This means that using the Robust Weight Similarity evolution has discovered profiles (at least one) whose variance is the least over all other approaches. However, this high quality profile might be the result of only a few similar sessions.

Figure 4.41: U of L Library: Evolution vs Traditional Variance Aggregates

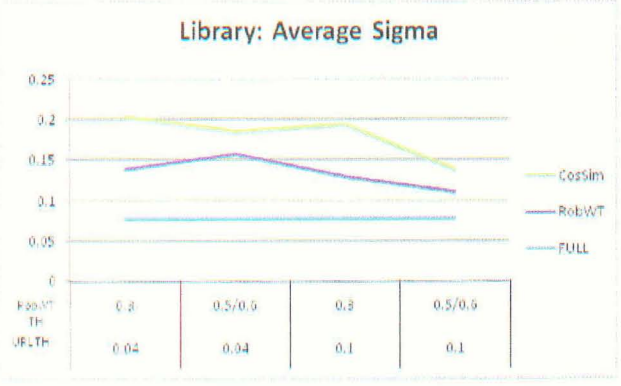


(a)

(b)



(c)



(d)

4.6 Experimental Design

An experimental design analysis was conducted on the University of Louisville dataset to find the significance of the configuration parameters on some of the evaluation metrics.

The factors used in experimental design analysis are listed in Table 4.6. For the "Discovery Approach" factor, the value 0 represents the "Evolution" mode, and 1 represents the "Traditional" discovery mode. The similarity matching method is encoded as 0 for the Binary Cosine Similarity and 1 for the Robust Weight Similarity. The similarity threshold has two levels: 0.3 and 0.55. During experiments, the value of 0.5 was used as the Binary Cosine Similarity threshold, whereas the value 0.6 was used for the Robust Weight Similarity threshold. However, to have a standardized view of the similarity threshold factor, the average of the two thresholds was used, so as to have only 2 levels instead of 3. The value of 0.05 is used as the sigma value σ.

Table 4.7 shows the response variables used in experimental design. Table 4.8 shows the data used in the experimental design analysis.

The results of experimental design, which are in Appendix A , showed that the URL weight threshold value and the profile discovery mode (evolution versus traditional) proved to be the most significant factors on profile variances and cardinalities (with small p-value), the similarity method (Cosine Binary versus Robust Weight) proved significant to some of the response variables (profile variances and densities), and the similarity threshold proved significant to some of the response variables (profile variances and number of profiles).

Table 4.6: Experimental Design Factors

Factor	Number of Levels	Level Values
Discovery Approach	2	0 , 1
Similarity Matching Method	2	0 , 1
URL Weight Threshold	2	0.04 , 0.1
Similarity Threshold	2	0.3 , 0.55

Table 4.7: Experimental Design Responses

Response	Level Values
Max Variance	[0-1]
Average variance	[0-1]
Number of Profiles	[1 - #sessions]
Cardinality % of Profile with Max Variance	$(0-1]$
Percentage of Duplicate Profiles	[0-1]
Profiles Density Average	[1-∞]

Table 4.8: Experimental Design Data Table

Disc. Approach	Sim. Match. Method	URL Wt Thr	Sim.Thr	Max var	Avg. Var	Num. Profiles	Card.%of Prf.with max var.	Dupl. %	Den. Avg.
0	0	0.04	0.30	0.966092	0.203	47	6.5	0.0120259	3553.6
0	0	0.04	0.55	0.961000	0.185	49	7.2	0.0026596	2442.2
0	0	0.10	0.30	0.435975	0.194	40	0.4	0.0102564	3467.3
0	0	0.10	0.55	0.428800	0.138	43	0.3	0.0066445	2975.0
0	1	0.04	0.30	0.460281	0.139	21	59.3	0.0285714	9380.6
0	1	0.04	0.55	0.962000	0.158	50	6.7	0.0048980	4996.6
0	1	0.10	0.30	0.448265	0.129	22	61.9	0.0519481	10186.8
0	1	0.10	0.55	0.475400	0.111	47	0.3	0.0120259	5671.9
1	0	0.04	0.30	0.203900	0.078	61	0.4	0.0018939	7867.0
1	0	0.04	0.55	0.203900	0.078	61	0.4	0.0018939	7867.0
1	0	0.10	0.30	0.203900	0.078	61	0.4	0.0018939	7867.0
1	0	0.10	0.55	0.203900	0.078	61	0.4	0.0018939	7867.0
1	1	0.04	0.30	0.203900	0.078	61	0.4	0.0018939	7867.0
1	1	0.04	0.55	0.203900	0.078	61	0.4	0.0018939	7867.0
1	1	0.10	0.30	0.203900	0.078	61	0.4	0.0018939	7867.0
1	1	0.10	0.55	0.203900	0.078	61	0.4	0.0018939	7867.0

4.7 Summary

Chapter 4 presented the results of applying the proposed framework on two datasets: University of Missouri CECS department's website logs, and University of Louisville's Library website logs. Each of the evaluation metrics and the proposed framework's parameters were discussed.

The results of the experiments showed that using the Robust Weight Similarity with a high similarity threshold value of 0.6, and a strict URL weight threshold of 0.1, gives the best results. The compactness of the profiles for this configuration kept improving over time, as seen in Figures 4.4 and 4.25. A lower profile variance means that the sessions are more similar to the profile. The higher quality of the profiles has made these profiles able to match more incoming new sessions, as seen in figures 4.7(d) and 4.28(d), where the majority of the profiles got updated. Moreover, this configuration resulted in more detailed profiles, where the average number of profile was around 20 when mining the University of Missouri CECS department's dataset and around 50 when mining the University of Louisville's Library dataset, as seen in Figures 4.14 and 4.35 respectively. These detailed profiles were also very different from each other (based on the profile pair-wise similarity measure), as seen in Figures 4.15(b) and 4.36(b), where the percentage of duplicate profiles was almost 0 for both cases. The profile quality using this configuration was further confirmed when the density of profiles was found to be increasing over time, as seen in Figures 4.17 and 4.38. The consistent increase in densities may be the ultimate judge that matching using the Robust Weights (which depend on accurate variance estimates) yields the best quality profiles compared to Cosine Similarity matching.

The results of experimental design showed that the URL weight threshold value and the profile discovery mode (evolution versus traditional) proved to be the most significant factors on profile variances and cardinalities (with small p-value), the similarity method (Cosine Binary versus Robust Weight) proved significant to some of

the response variables (profile variances and densities), and the similarity threshold proved significant to some of the response variables (profile variances and number of profiles).

CHAPTER 5

CONCLUSION

The increasing importance of the user has challenged online organizations to start tracking the behavior of their website users on the web, in order to better understand and satisfy their needs, and hence, maintain the loyalty of current users and lure new users. Since web usage data tends to be huge, heterogeneous, and continuously growing and changing, traditional data management tools are neither feasible nor cost-effective to handle them. Web usage mining is the process of discovering interesting usage behavior over the World Wide Web. Many studies have been conducted to adapt web usage mining to business needs, however, very few have tried to handle the changing nature of the web user activities.

This thesis has presented an innovative and scalable framework that is capable of capturing the changing behavior of users over the World Wide Web. The proposed framework develops a set of evolving profiles that represent the usage pattern as a set of URLs with varying degrees of significance. These profiles can be considered as representatives of usage behavior at any given time. The proposed framework is applied each time that new data becomes available. It tries to match these transactions with previous user behavior (as in collaborative filtering), uses matching transactions to update the old profiles, and subjects new (non-matching) transactions to a new pattern discovery process using the Hierarchical Unsupervised Niche Clustering (HUNC)

algorithm.

The resulting profiles are formulated in way that makes it intuitive to utilize them in higher-level applications to meet business goals such as web recommendations, web personalization, and user-driven web content. These evolving profiles also allow a better understanding of the changing interests of visitors to a website.

The proposed framework was applied on the web server logs of two real web sites: University of Missouri's CECS department website, and University of Louisville's Library website. The developed usage profiles were evaluated using a set of metrics including the profile variance (how much the sessions in a cluster are dispersed or different from each other), profile cardinality (how many sessions the profile represents), the percentage of matching transactions (to which degree are the new transactions similar to older profiles), profiles' pair-wise similarity (how similar the developed profiles are to each other), the profile density (which is the ratio of the profile variance and cardinality), and the total number of profiles and their types (new, distinct, or static).

To find high quality profiles, several parameters were varied within different configurations. These parameters include the matching method which could be the Binary Cosine Similarity or the Robust Weight Similarity (which is variance-sensitive), the similarity threshold value which controls how strict this matching is, and the URL significance weight threshold which controls the quality of URLs allowed in the profile. Moreover, the profiles resulting from the evolution mode of discovery were compared to the traditional mode, where all the available data is mined in one shot.

The experiments have showed that the output quality should be evaluated by scrutinizing the different parameter configurations, and observing the evaluation metrics from different perspectives, because there is always a trade-off between different quality measures. For example, an increase in the profile cardinality (number of sessions assigned to the profile) might cause a decrease in the profile's compactness (i.e. higher

variance), which may not be desirable.

The results of the experiments also showed that using the Robust Weight Similarity with a high similarity threshold value of 0.6, and a strict URL weight threshold of 0.1, gives the best results. The compactness (variance) of the profiles for this configuration kept improving over time, the higher quality of the profiles has made them match more incoming new sessions, more detailed profiles were discovered, and those detailed profiles were also very different from each other (as verified by the low profile pair-wise similarity measures). The profile quality using this configuration was further confirmed when the density of profiles was found to be increasing over time. The consistent increase in densities may be the ultimate judge that matching using the Robust Weights (which depend on accurate variance estimates) yields the best quality profiles compared to Cosine Similarity matching.

The proposed framework's output patterns were comparable to the ones resulting from the traditional pattern discovery mode. The latter can be considered as the best output possible since all usage data is mined at once (instead of at different time periods as in the proposed approach). Having said that, the proposed framework has still shown desired behavior and resulted in high quality profiles that are good representatives of the users' browsing behavior at any given time. Most importantly, the proposed framework has the critical advantage of enabling scalability in handling very large usage data that makes it impossible to mine all patterns in one shot.

The results of experimental design showed that the URL weight threshold value and the profile discovery mode (evolution versus traditional) proved to be the most significant factors on profile variances and cardinalities (with small p-value), the similarity method (Cosine Binary versus Robust Weight) proved significant to some of the response variables (profile variances and densities), and the similarity threshold proved significant to some of the response variables (profile variances and number of profiles).

This thesis has achieved its objectives, and succeeded in providing the means to track usage changes over the web. However, this framework can be improved further. As part of future work, experiments on more challenging datasets could be conducted, and more parameters, such as HUNC parameters, could be studied in the evaluation experiments. A forgetting factor could also be considered so that new data would have more importance compared to older data as in [7]. Also, developing a high-level application, such as a recommendation system, that uses the generated profiles could give more insight when evaluating the performance of proposed framework.

REFERENCES

[1] R. Cooley, B. Mobasher, and J. Srivastava. Web Mining: Information and Pattern discovery on the World Wide Web, Proc. IEEE Intl. Conf. Tools with AI, Newport Beach, CA, pp. 558-567, 1997.

[2] O. Nasraoui and R. Krishnapuram, and A. Joshi. Mining Web Access Logs Using a Relational Clustering Algorithm Based on a Robust Estimator, 8th International World Wide Web Conference, Toronto, pp. 40-41, 1999.

[3] J. Srivastava, R. Cooley, M. Deshpande. and P-N Tan, Web usage mining: Discovery and applications of usage patterns from web data, SIGKDD Explorations, Vol. 1, No. 2, Jan 2000, pp. 1-12.

[4] M. Spiliopoulou and L. C. Faulstich. WUM: A Web utilization Miner, in Proceedings of EDBT workshop WebDB98, Valencia, Spain, 1999.

[5] T. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal. From user access patterns to dynamic hypertext linking. In Proceedings of the 5th International World Wide Web conference, Paris, France, 1996.

[6] O. Nasraoui and R. Krishnapuram. A New Evolutionary Approach to Web Usage and Context Sensitive Associations Mining, International Journal on Computational Intelligence and Applications - Special Issue on Internet Intelligent Systems, Vol. 2, No. 3, pp. 339-348, Sep. 2002.

[7] O. Nasraoui, C. Cardona, C. Rojas, and F. Gonzalez. Mining Evolving User Profiles in Noisy Web Clickstream Data with a Scalable Immune System Clustering Algorithm, in Proc. of WebKDD 2003, Washington DC, Aug. 2003, 71-81.

[8] P. Desikan and J. Srivastava, Mining Temporally Evolving Graphs. In Proceedings of "WebKDD- 2004 workshop on Web Mining and Web Usage Analysis", B. Mobasher, B. Liu, B. Masand, O. Nasraoui, Eds. part of the ACM KDD: Knowledge Discovery and Data Mining Conference, Seattle, WA, 2004.

[9] O. Nasraoui, C. Rojas, and C. Cardona, A Framework for Mining Evolving Trends in Web Data Streams using Dynamic Learning and Retrospective Validation, in Computer Networks, Special Issue on Web Dynamics, 50(14), Oct., 2006.

[10] I. Grabtree and S. Soltysiak, Identifying and Tracking Changing Interests. International Journal of Digital Libraries, vol. 2, 38-53.

[11] I. Koychev, Gradual Forgetting for Adaptation to Concept Drift. In Proceedings of ECAI 2000 Workshop Current Issues in Spatio-Temporal Reasoning. Berlin, Germany, 2000, pp. 101-106

[12] O. Nasraoui, M. Soliman, E. Saka, A. Badia, R. Germain, A Web Usage Mining Framework for Mining Evolving user Profiles in Dynamic web Sites, IEEE Transactions on Knowledge and Data Engineering, Feb. 2008, pp 202-215

[13] A. Maloof, S. Michalski, Learning Evolving Concepts Using Partial-Memory Approach. Working Notefs of the 1995 AAAI Fall Symposium on Active learning, Boston, MA. Pp 70-73

[14] Nasraoui O., and Krishnapuram R., "A Novel Approach to Unsupervised Robust Clustering using Genetic Niching," Proc. of the 9th IEEE International Conf. on Fuzzy Systems, San Antonio, TX, May 2000, pp. 170-175.

[15] L. Catledge and J. Pitkow. Characterizing browsing behaviors on the World Wide Web. Computer Networks and ISDN Systems, 27(6), 1995.

[16] J. H. Holland, Adaption in Natural and Artificial Systems (MIT Press, 1975)

[17] O. Nasraoui, C. Cardona-Uribe, C. Rojas-Coronel, "TECNO-Streams": tracking evolving clusters in noisy data streams with a scalable immune system learning model, in: IEEE International Conference on Data Mining, Melbourne, Florida, November 2003.

[18] J. Pitkow and K.Bharat. Webviz: A tool for world-wide web access log analysis. In First International WWW Conference, 1994.

[19] B. Mobasher, N. Jain, E.Han, and J. Srivastava. Web mining: Pattern discovery from world wide web transactions. Technical Report TR 96-050, University of Minnesota, Dept. of Computer Science, Minneapolis, 1996

[20] R. Sarukkai, Link prediction and path analysis using Markov chains, Computer Networks Volume 33, Issues 1-6, June 2000, Pages 377-386

[21] R. Agrawal and R. Srikant, "Mining sequential patterns," p. 3, 11th International Conference on Data Engineering (ICDE'95), 1995

[22] P.-N. Tan, M. Steinbach, and V. Kumar, Introduction to Data Mining, Addison Wesley, 2005.

[23] R.Baraglia and F. Silvestri, "An Online Recommender System for Large Web Sites" IEEE/WIC/ACM International Conference on Web Intelligence (WI'04), pp. 199-205, 2004.

[24] Agrawal, R. and Srikant, R. Fast Algorithms for Mining Association Rules in Large Databases. In Proceedings of the 20th international Conference on Very Large Data Bases, p. 487-499, Sep. 1994.

[25] J Borges and M Levene , Data Mining of User Navigation Patterns, in Proc. of WebKDD 2000, KDD workshop on Web Usage Analysis and User Profiling, 2000

[26] M Spiliopoulou, C Pohle, and LC Faulstich, Improving the Effectiveness of a Web Site with Web Usage Mining, In Proc. of WebKDD 2000, KDD workshop on Web Usage Analysis and User Profiling, 2000.

APPENDIX A

Max Variance:

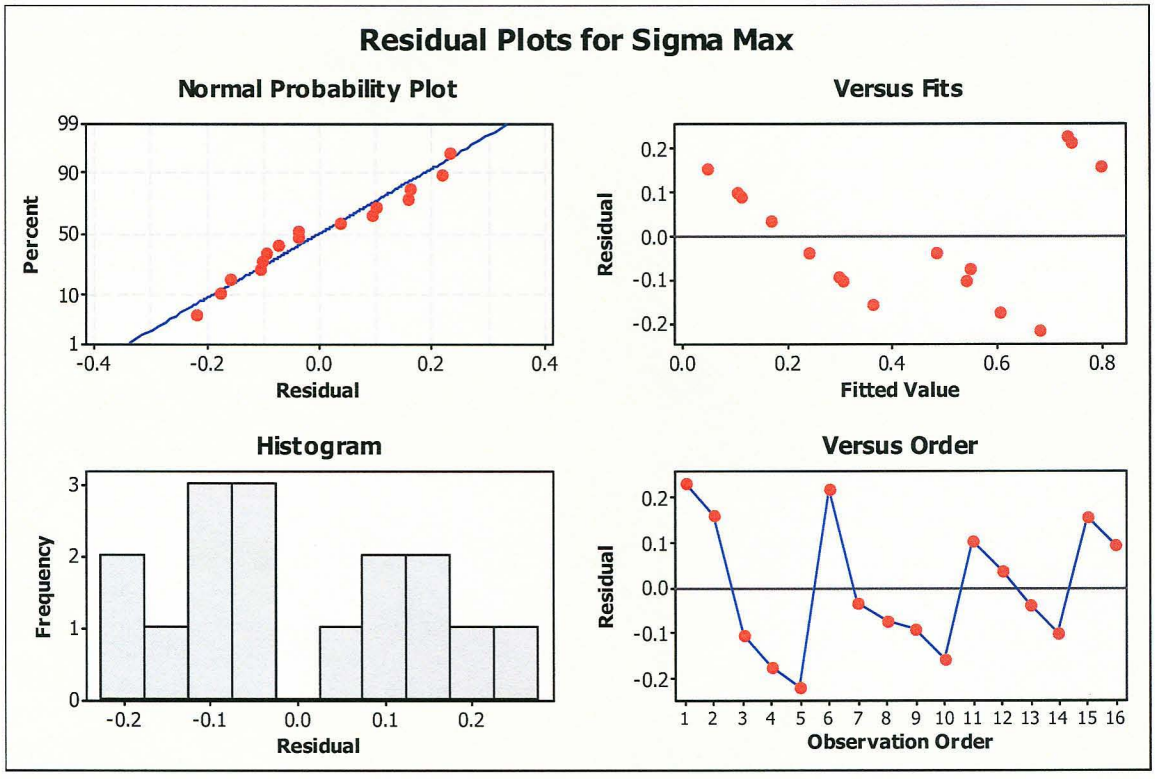
General Linear Model: Sigma Max versus Method, Matching Method, ...

Factor	Type	Levels	Values
Method	fixed	2	0, 1
Matching Method	fixed	2	0, 1
URL	fixed	2	0.04, 0.10
Matching Sim	fixed	2	0.30, 0.55

Analysis of Variance for Sigma Max, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Method	1	0.76852	0.76852	0.76852	26.92	0.000
Matching Method	1	0.01243	0.01243	0.01243	0.44	0.523
URL	1	0.15228	0.15228	0.15228	5.33	0.041
Matching Sim	1	0.01668	0.01668	0.01668	0.58	0.461
Error	11	0.31402	0.31402	0.02855		
Total	15	1.26393				

S = 0.168959 R-Sq = 75.16% R-Sq(adj) = 66.12%



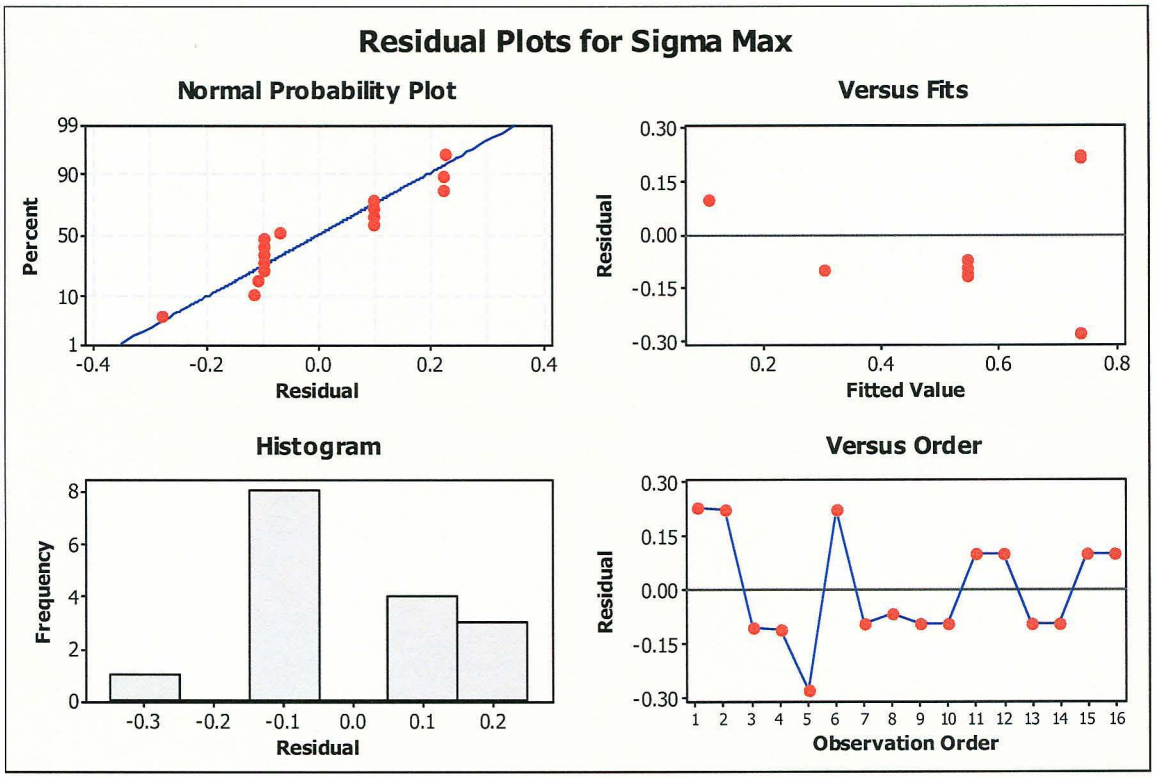
General Linear Model: Sigma Max versus Method, URL

Factor	Type	Levels	Values
Method	fixed	2	0, 1
URL	fixed	2	0.04, 0.10

Analysis of Variance for Sigma Max, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Method	1	0.76852	0.76852	0.76852	29.12	0.000
URL	1	0.15228	0.15228	0.15228	5.77	0.032
Error	13	0.34313	0.34313	0.02639		
Total	15	1.26393				

S = 0.162463 R-Sq = 72.85% R-Sq(adj) = 68.68%



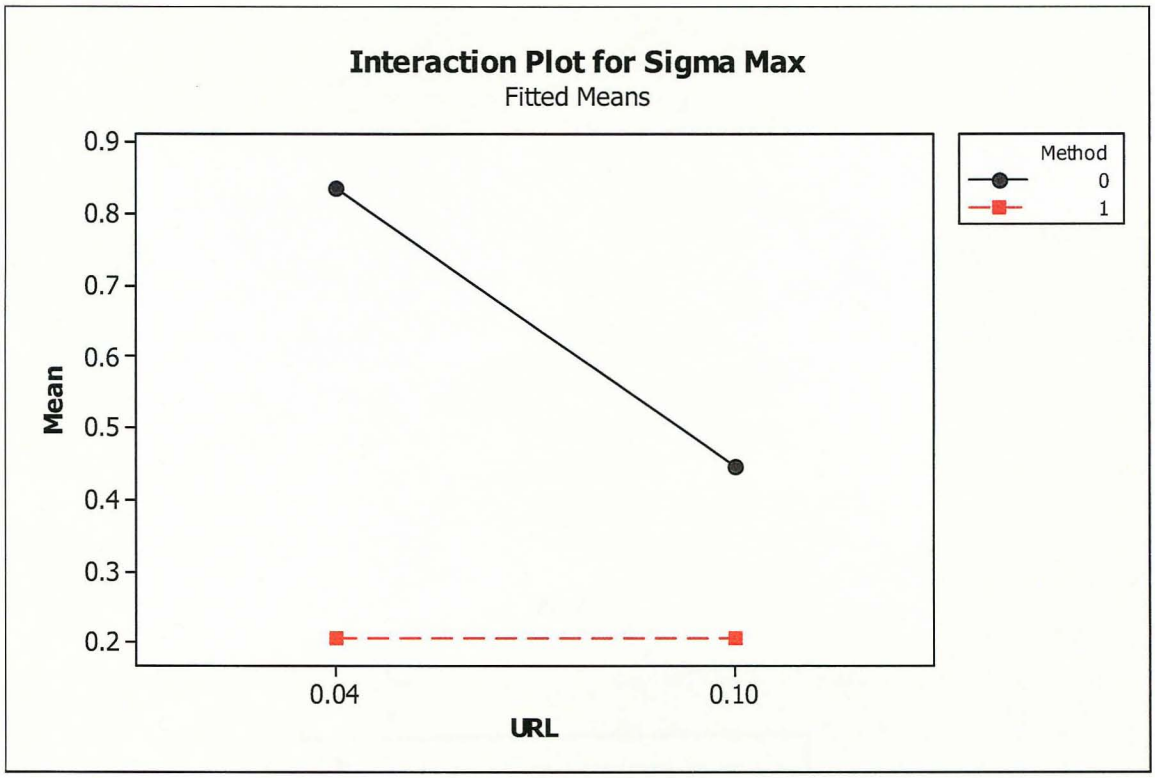
General Linear Model: Sigma Max versus Method, URL

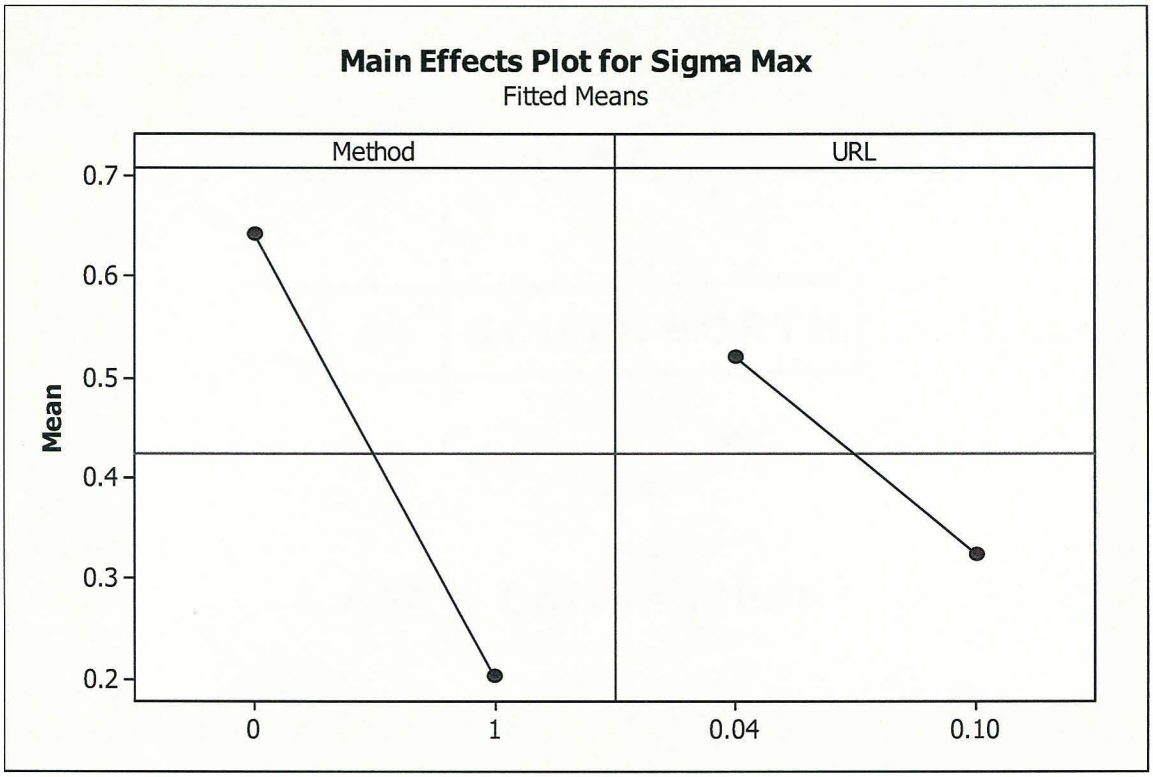
Factor	Type	Levels	Values
Method	fixed	2	0, 1
URL	fixed	2	0.04, 0.10

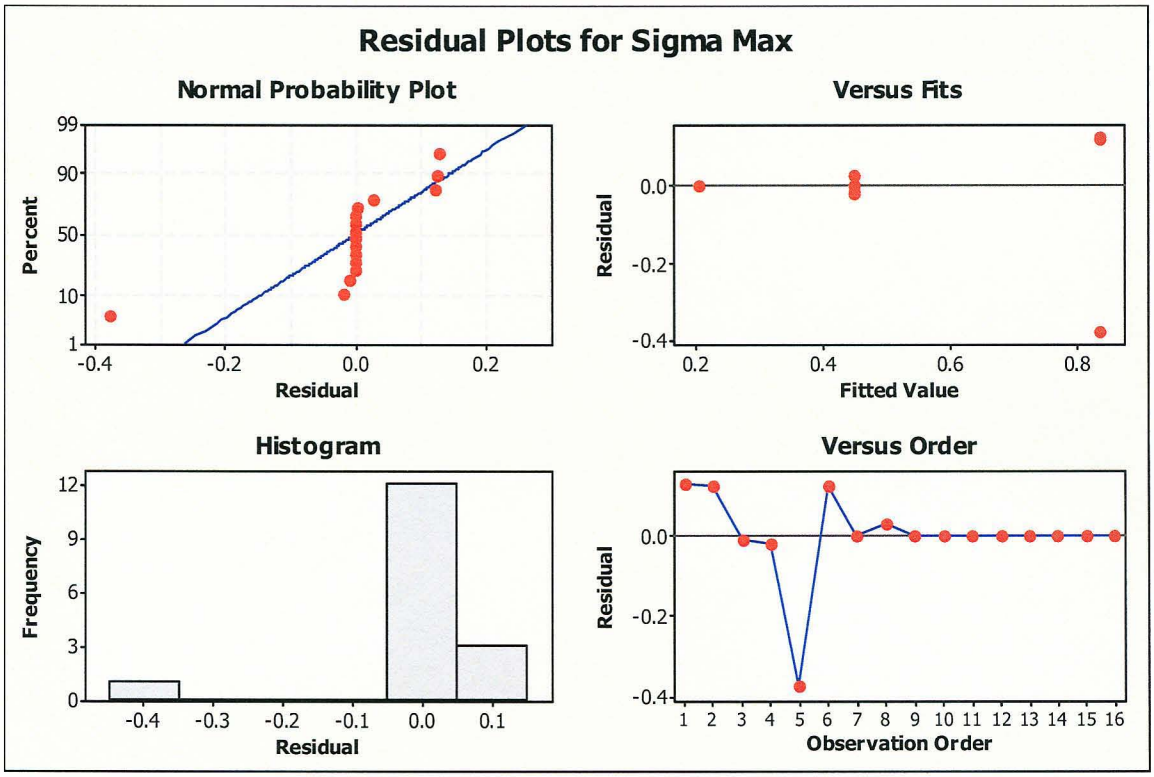
Analysis of Variance for Sigma Max, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Method	1	0.76852	0.76852	0.76852	48.32	0.000
URL	1	0.15228	0.15228	0.15228	9.58	0.009
Method*URL	1	0.15228	0.15228	0.15228	9.58	0.009
Error	12	0.19084	0.19084	0.01590		
Total	15	1.26393				

S = 0.126110 R-Sq = 84.90% R-Sq(adj) = 81.13%




Main Effects Plot for Sigma Max
Fitted Means
Method
URL
Mean
0.7
0.6
0.5
0.4
0.3
0.2
0
1
0.04
0.10


Residual Plots for Sigma Max
Normal Probability Plot
Percent
99
90
50
10
1
-0.4
-0.2
0.0
0.2
Residual
Versus Fits
Residual
0.0
-0.2
-0.4
0.2
0.4
0.6
0.8
Fitted Value
Histogram
Frequency
12
9
6
3
0
-0.4
-0.3
-0.2
-0.1
0.0
0.1
Residual
Versus Order
Residual
0.0
-0.2
-0.4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Observation Order

Variance Average:

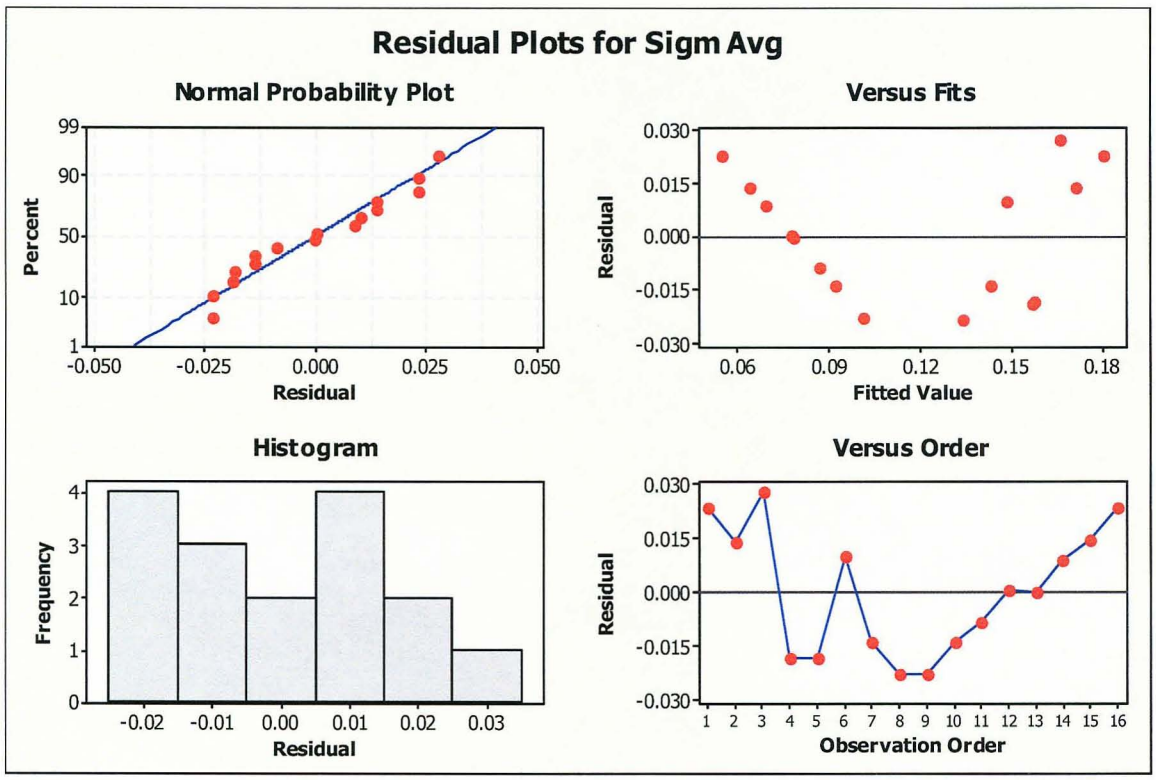
General Linear Model: Sigm Avg versus Method, Matching Method, ...

Factor	Type	Levels	Values
Method	fixed	2	0, 1
Matching Method	fixed	2	0, 1
URL	fixed	2	0.04, 0.10
Matching Sim	fixed	2	0.30, 0.55

Analysis of Variance for Sigm Avg, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Method	1	0.0252565	0.0252565	0.0252565	59.62	0.000
Matching Method	1	0.0021011	0.0021011	0.0021011	4.96	0.048
URL	1	0.0008100	0.0008100	0.0008100	1.91	0.194
Matching Sim	1	0.0003383	0.0003383	0.0003383	0.80	0.391
Error	11	0.0046596	0.0046596	0.0004236		
Total	15	0.0331655				

S = 0.0205815 R-Sq = 85.95% R-Sq(adj) = 80.84%



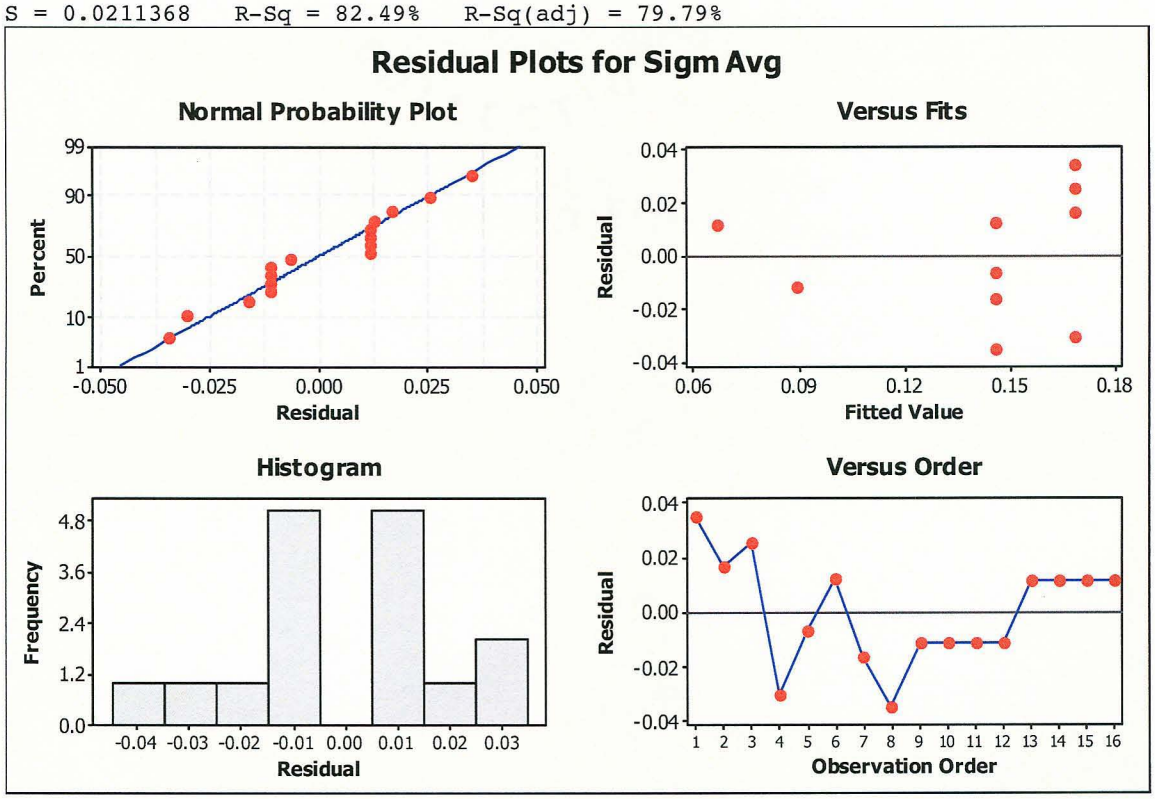
General Linear Model: Sigm Avg versus Method, Matching Method

Factor	Type	Levels	Values
Method	fixed	2	0, 1
Matching Method	fixed	2	0, 1

Analysis of Variance for Sigm Avg, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Method	1	0.025256	0.025256	0.025256	56.53	0.000
Matching Method	1	0.002101	0.002101	0.002101	4.70	0.049
Error	13	0.005808	0.005808	0.000447		
Total	15	0.033165				

S = 0.0211368 R-Sq = 82.49% R-Sq(adj) = 79.79%



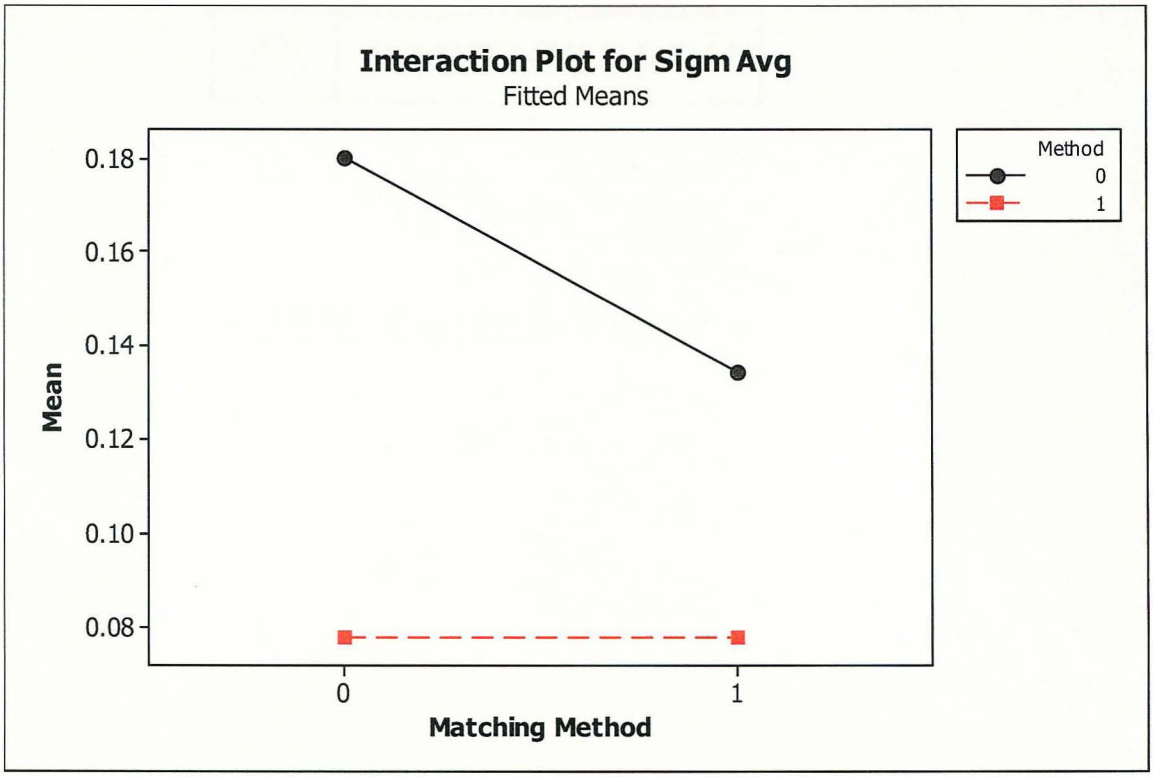
General Linear Model: Sigm Avg versus Method, Matching Method

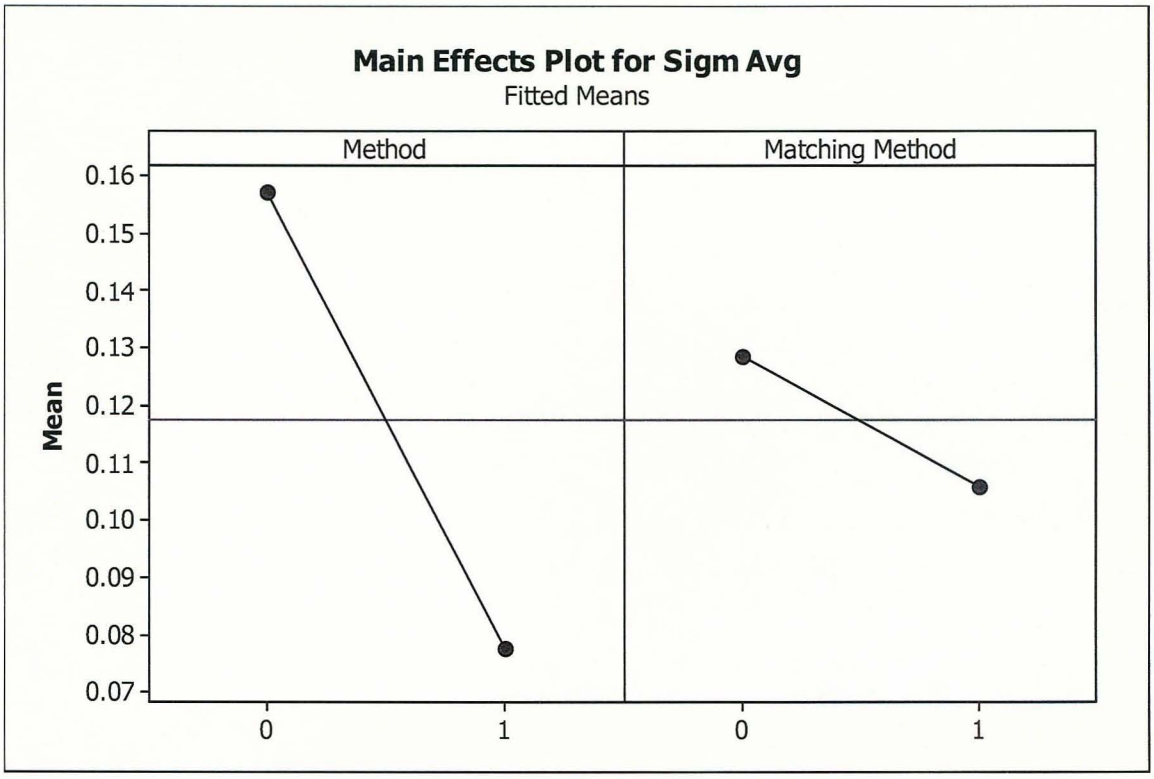
Factor	Type	Levels	Values
Method	fixed	2	0, 1
Matching Method	fixed	2	0, 1

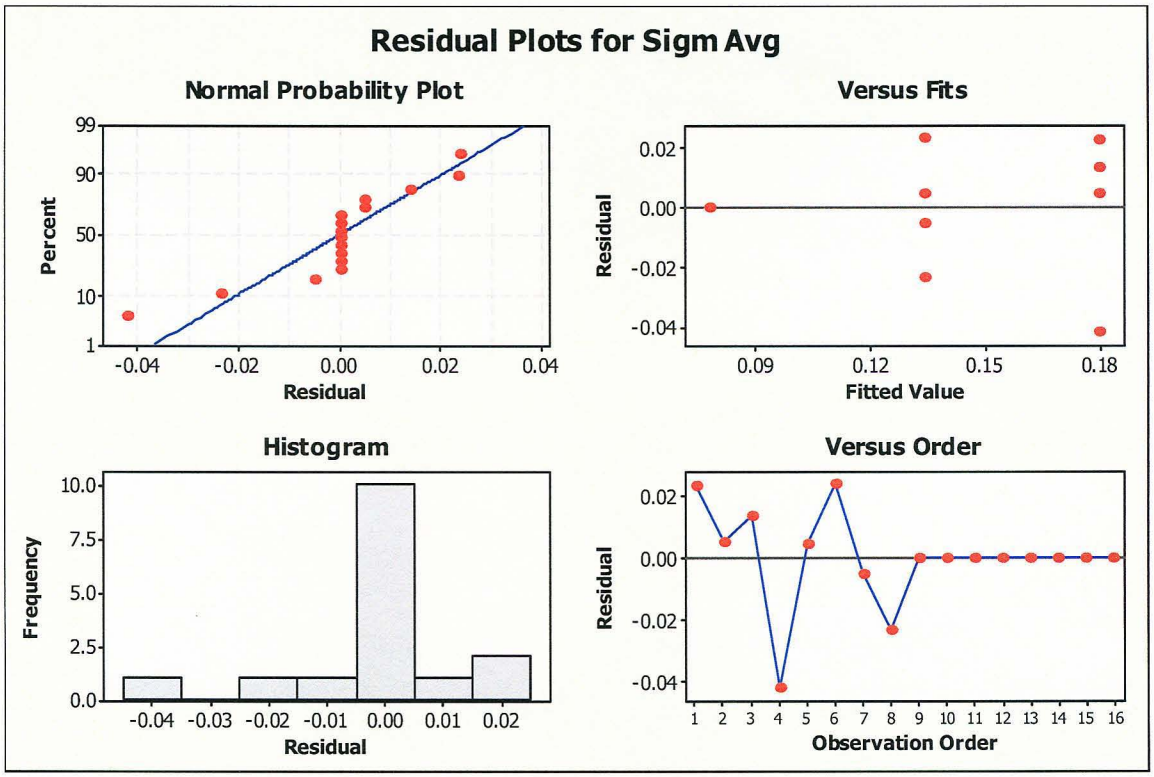
Analysis of Variance for Sigm Avg, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Method	1	0.0252565	0.0252565	0.0252565	81.76	0.000
Matching Method	1	0.0021011	0.0021011	0.0021011	6.80	0.023
Method*Matching Method	1	0.0021011	0.0021011	0.0021011	6.80	0.023
Error	12	0.0037069	0.0037069	0.0003089		
Total	15	0.0331655				

S = 0.0175757 R-Sq = 88.82% R-Sq(adj) = 86.03%


Interaction Plot for Sigm Avg
Fitted Means
Method
0
1
Mean
0.18
0.16
0.14
0.12
0.10
0.08
0
1
Matching Method


Main Effects Plot for Sigm Avg
Fitted Means
Method
Matching Method
Mean
0.16
0.15
0.14
0.13
0.12
0.11
0.10
0.09
0.08
0.07
0
1
0
1



Profiles Counts:

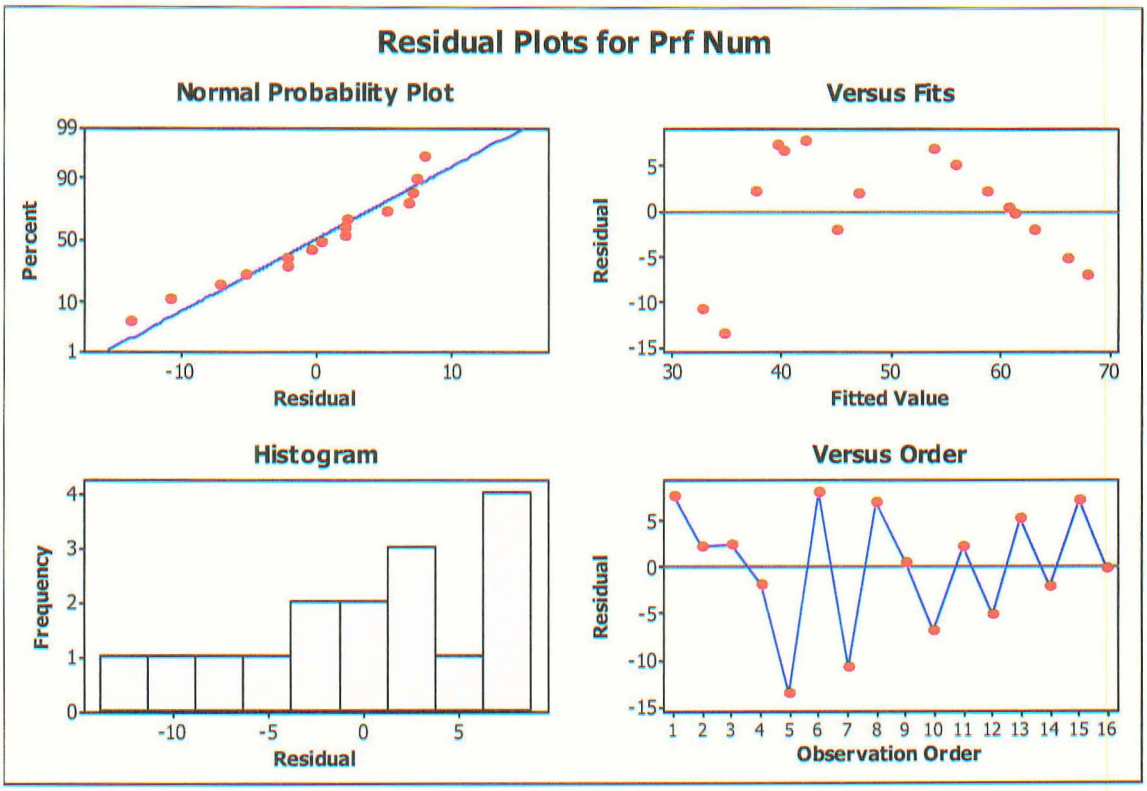
General Linear Model: Prf Num versus Method, Matching Method, ...

```
Factor           Type   Levels  Values
Method           fixed       2  0, 1
Matching Method  fixed       2  0, 1
URL              fixed       2  0.04, 0.10
Matching Sim     fixed       2  0.30, 0.55
```

Analysis of Variance for Prf Num, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Method	1	1785.06	1785.06	1785.06	30.39	0.000
Matching Method	1	95.06	95.06	95.06	1.62	0.230
URL	1	14.06	14.06	14.06	0.24	0.634
Matching Sim	1	217.56	217.56	217.56	3.70	0.081
Error	11	646.19	646.19	58.74		
Total	15	2757.94				

S = 7.66448 R-Sq = 76.57% R-Sq(adj) = 68.05%



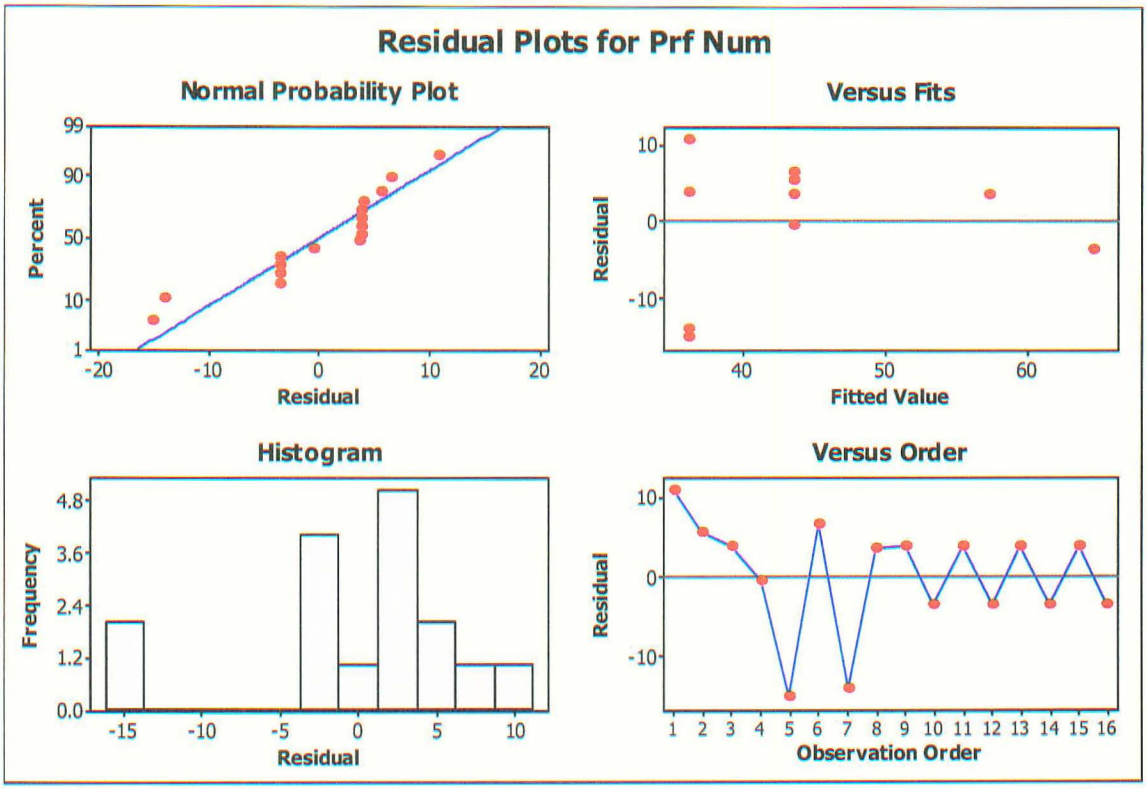
General Linear Model: Prf Num versus Method, Matching Sim

```
Factor         Type   Levels  Values
Method         fixed       2  0, 1
Matching Sim   fixed       2  0.30, 0.55


Analysis of Variance for Prf Num, using Adjusted SS for Tests

Source         DF  Seq SS  Adj SS  Adj MS      F      P
Method          1  1785.1  1785.1  1785.1  30.72  0.000
Matching Sim    1   217.6   217.6   217.6   3.74  0.075
Error          13   755.3   755.3    58.1
Total          15  2757.9


S = 7.62240   R-Sq = 72.61%   R-Sq(adj) = 68.40%
```



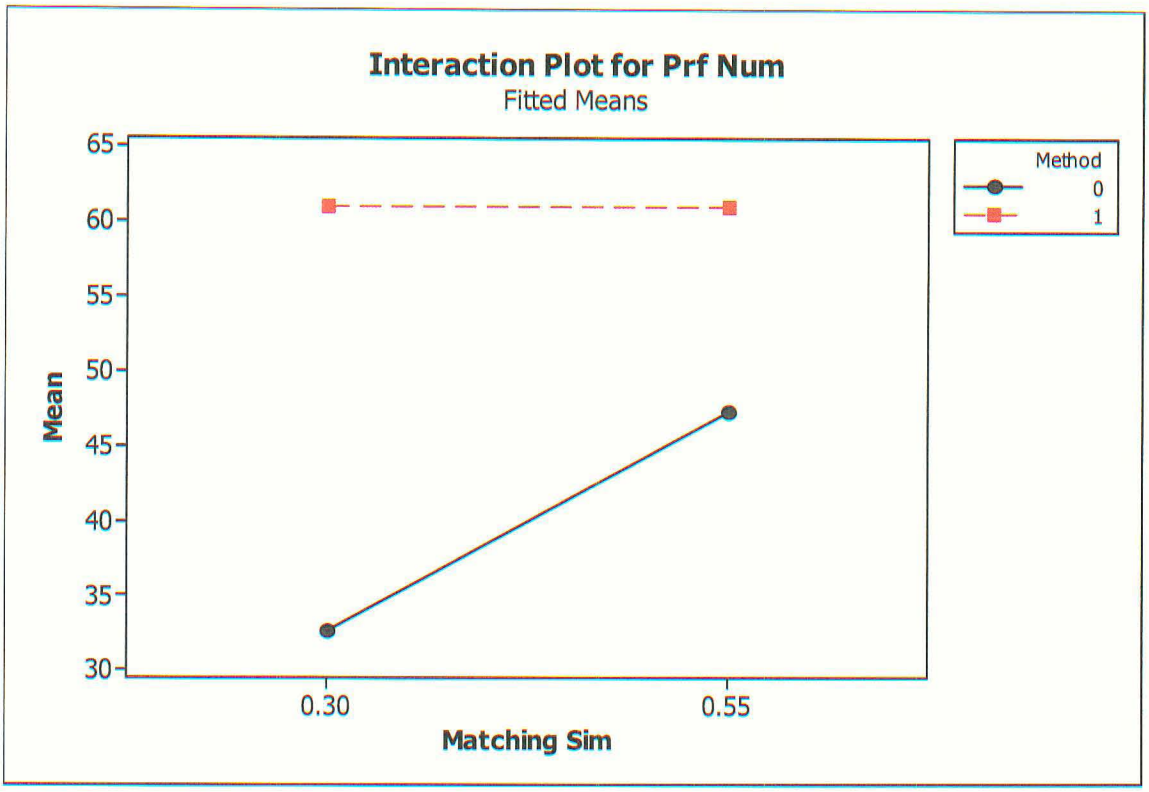
General Linear Model: Prf Num versus Method, Matching Sim

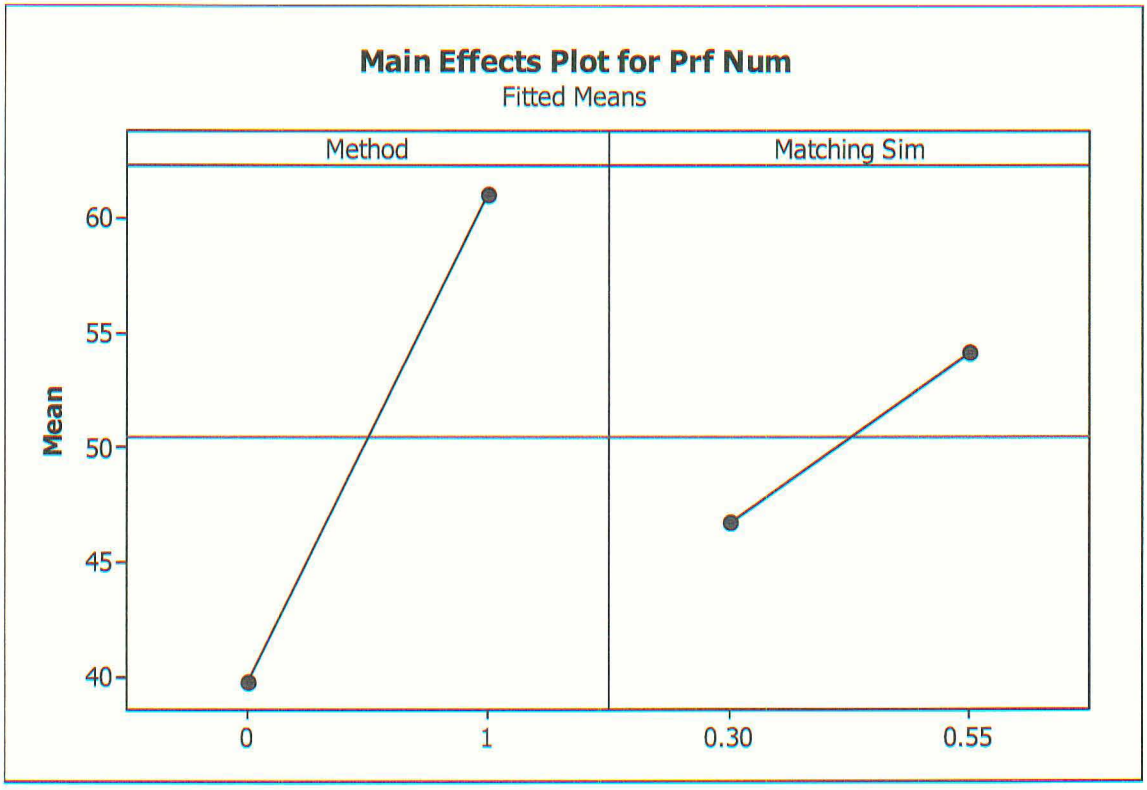
Factor	Type	Levels	Values
Method	fixed	2	0, 1
Matching Sim	fixed	2	0.30, 0.55

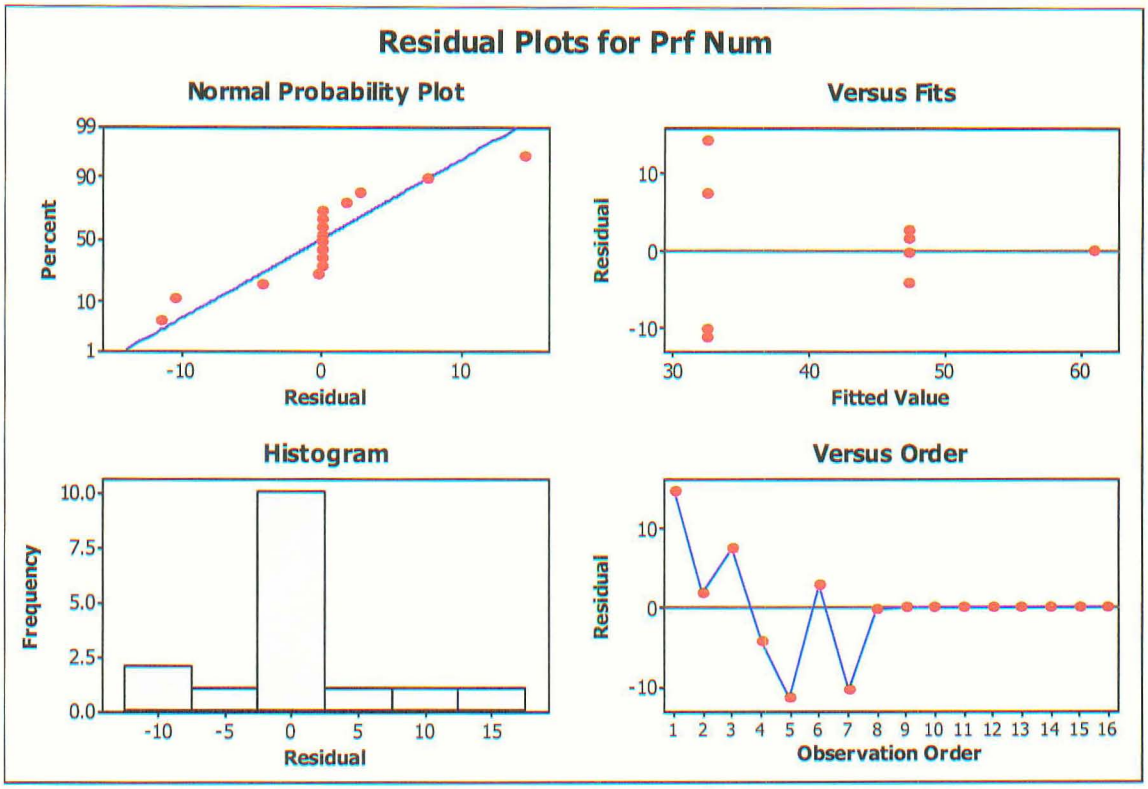
Analysis of Variance for Prf Num, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Method	1	1785.06	1785.06	1785.06	39.83	0.000
Matching Sim	1	217.56	217.56	217.56	4.85	0.048
Method*Matching Sim	1	217.56	217.56	217.56	4.85	0.048
Error	12	537.75	537.75	44.81		
Total	15	2757.94				

S = 6.69421 R-Sq = 80.50% R-Sq(adj) = 75.63%


Interaction Plot for Prf Num
Fitted Means
Mean
65
60
55
50
45
40
35
30
0.30
0.55
Matching Sim
Method
0
1


Main Effects Plot for Prf Num
Fitted Means
Method
Matching Sim
Mean
60
55
50
45
40
0
1
0.30
0.55



Cardinality of profile with maximum sigma:

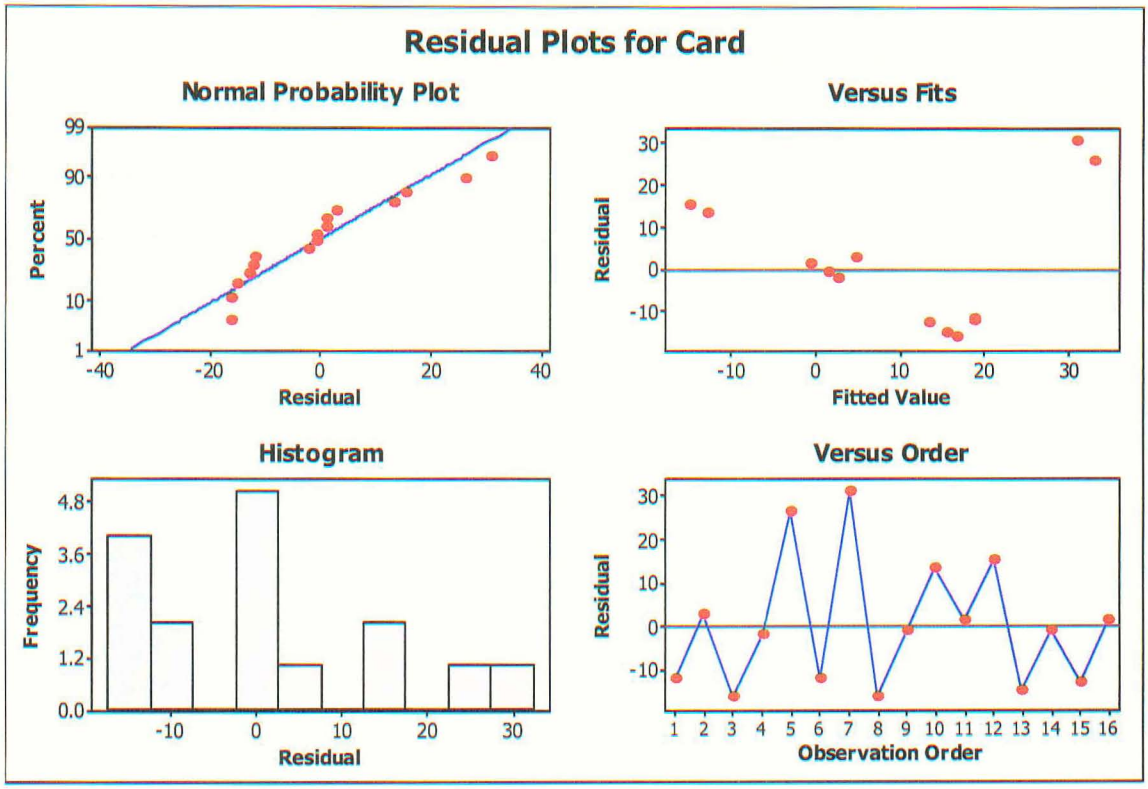
General Linear Model: Card versus Method, Matching Method, ...

Factor	Type	Levels	Values
Method	fixed	2	0, 1
Matching Method	fixed	2	0, 1
URL	fixed	2	0.04, 0.10
Matching Sim	fixed	2	0.30, 0.55

Analysis of Variance for Card, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Method	1	1221.3	1221.3	1221.3	4.06	0.069
Matching Method	1	808.3	808.3	808.3	2.68	0.130
URL	1	17.4	17.4	17.4	0.06	0.814
Matching Sim	1	806.8	806.8	806.8	2.68	0.130
Error	11	3312.5	3312.5	301.1		
Total	15	6166.4				

S = 17.3532 R-Sq = 46.28% R-Sq(adj) = 26.75%



General Linear Model: Card versus Method, Matching Method, Matching Sim

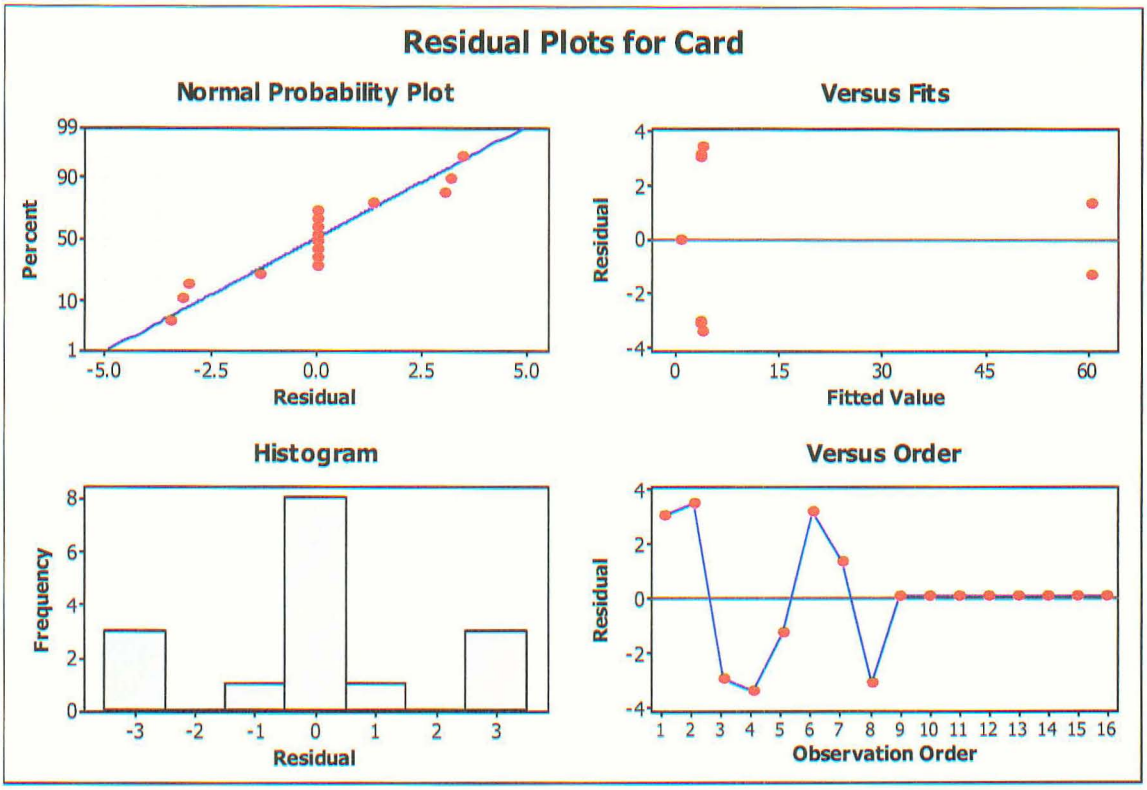
Factor	Type	Levels	Values
Method	fixed	2	0, 1
Matching Method	fixed	2	0, 1
Matching Sim	fixed	2	0.30, 0.55

Analysis of Variance for Card, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F
Method	1	1221.34	1221.34	1221.34	147.68
Matching Method	1	808.31	808.31	808.31	97.74
Matching Sim	1	806.83	806.83	806.83	97.56
Method*Matching Method	1	808.31	808.31	808.31	97.74
Method*Matching Sim	1	806.83	806.83	806.83	97.56
Matching Method*Matching Sim	1	824.30	824.30	824.30	99.67
Method*Matching Method*Matching Sim	1	824.30	824.30	824.30	99.67
Error	8	66.16	66.16	8.27	
Total	15	6166.38			

Source	P
Method	0.000
Matching Method	0.000
Matching Sim	0.000
Method*Matching Method	0.000
Method*Matching Sim	0.000
Matching Method*Matching Sim	0.000
Method*Matching Method*Matching Sim	0.000
Error	

S = 2.87580 R-Sq = 98.93% R-Sq(adj) = 97.99%



Number of duplicate profiles:

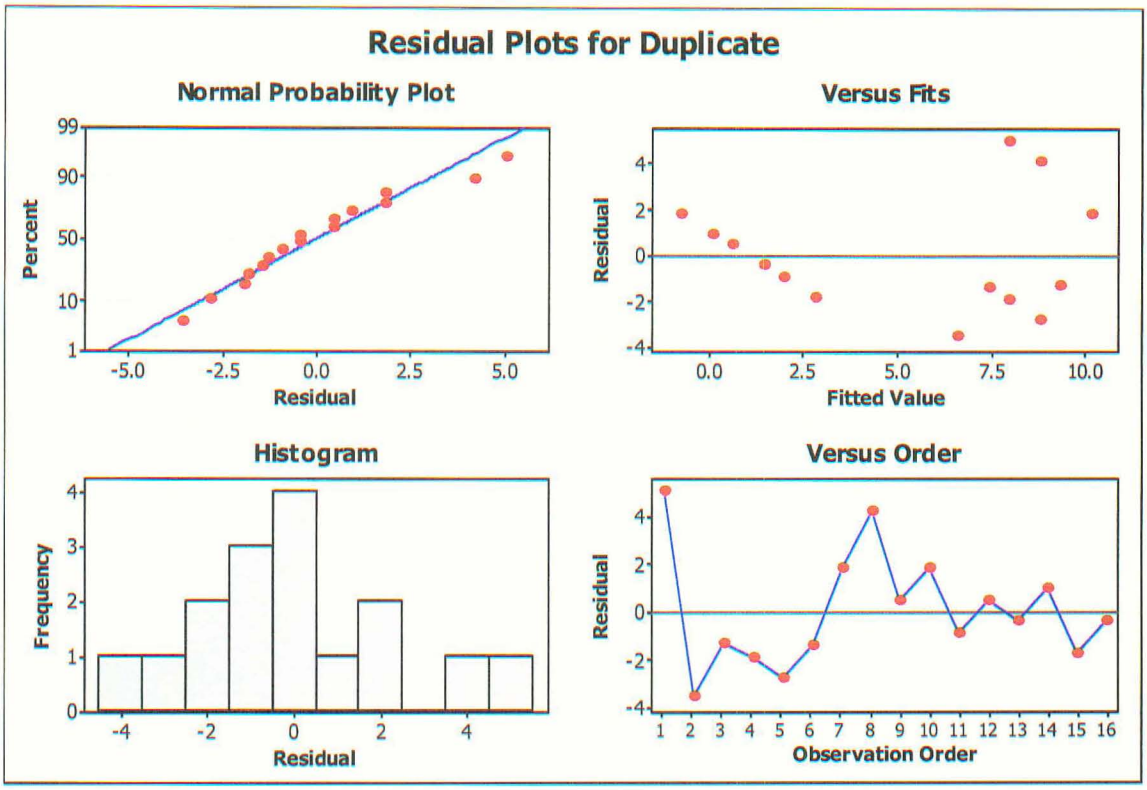
General Linear Model: Duplicate versus Method, Matching Method, ...

Factor	Type	Levels	Values
Method	fixed	2	0, 1
Matching Method	fixed	2	0, 1
URL	fixed	2	0.04, 0.10
Matching Sim	fixed	2	0.30, 0.55

Analysis of Variance for Duplicate, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Method	1	0.0008105	0.0008105	0.0008105	7.44	0.020
Matching Method	1	0.0002711	0.0002711	0.0002711	2.49	0.143
URL	1	0.0000669	0.0000669	0.0000669	0.61	0.450
Matching Sim	1	0.0003665	0.0003665	0.0003665	3.36	0.094
Error	11	0.0011990	0.0011990	0.0001090		
Total	15	0.0027140				

S = 0.0104405 R-Sq = 55.82% R-Sq(adj) = 39.76%



General Linear Model: Duplicate versus Method, Matching Method, ...

Factor	Type	Levels	Values
Method	fixed	2	0, 1
Matching Method	fixed	2	0, 1
URL	fixed	2	0.04, 0.10
Matching Sim	fixed	2	0.30, 0.55

Analysis of Variance for Duplicate, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Method	1	0.0008105	0.0008105	0.0008105	**	
Matching Method	1	0.0002711	0.0002711	0.0002711	**	
URL	1	0.0000669	0.0000669	0.0000669	**	
Matching Sim	1	0.0003665	0.0003665	0.0003665	**	
Method*Matching Method	1	0.0002711	0.0002711	0.0002711	**	
Method*URL	1	0.0000669	0.0000669	0.0000669	**	
Method*Matching Sim	1	0.0003665	0.0003665	0.0003665	**	
Matching Method*URL	1	0.0000500	0.0000500	0.0000500	**	
Matching Method*Matching Sim	1	0.0001601	0.0001601	0.0001601	**	
URL*Matching Sim	1	0.0000069	0.0000069	0.0000069	**	
Method*Matching Method*URL	1	0.0000500	0.0000500	0.0000500	**	
Method*Matching Method*Matching Sim	1	0.0001601	0.0001601	0.0001601	**	
Method*URL*Matching Sim	1	0.0000069	0.0000069	0.0000069	**	
Matching Method*URL*Matching Sim	1	0.0000303	0.0000303	0.0000303	**	
Method*Matching Method*URL* Matching Sim	1	0.0000303	0.0000303	0.0000303	**	
Error	0	*	*	*		
Total	15	0.0027140				

Profiles Density:

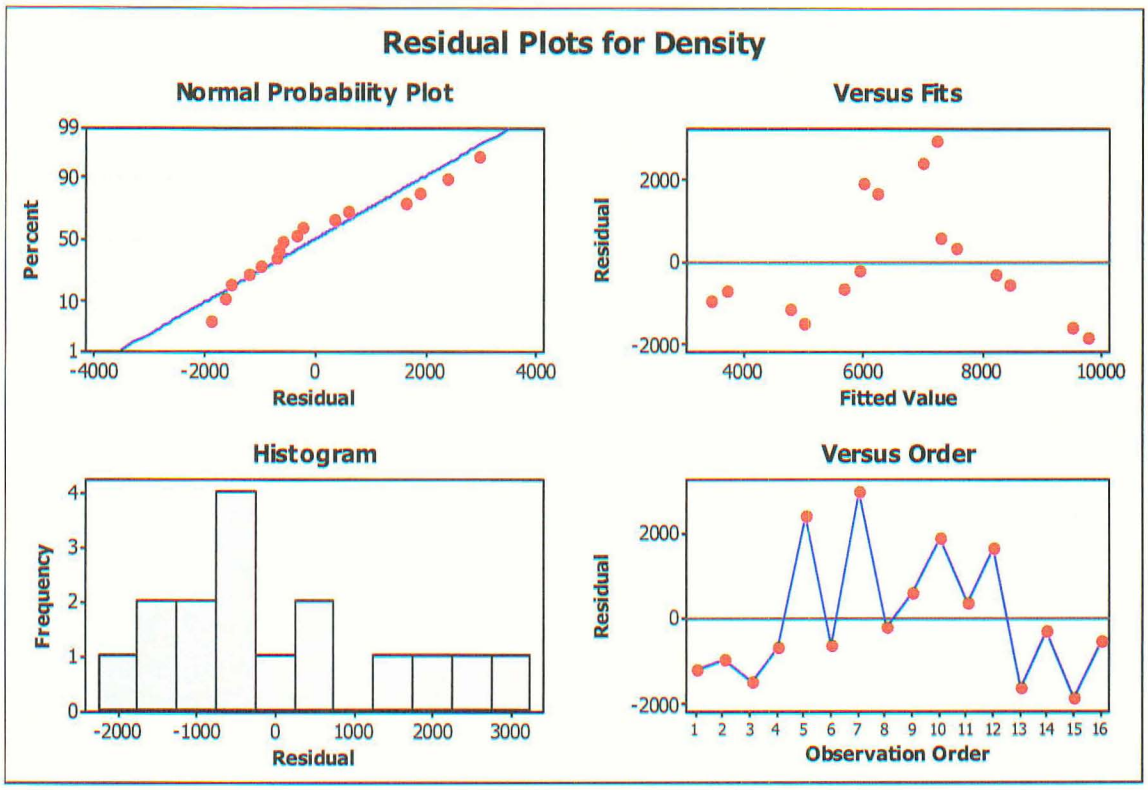
General Linear Model: Density versus Method, Matching Method, ...

Factor	Type	Levels	Values
Method	fixed	2	0, 1
Matching Method	fixed	2	0, 1
URL	fixed	2	0.04, 0.10
Matching Sim	fixed	2	0.30, 0.55

Analysis of Variance for Density, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Method	1	25658023	25658023	25658023	8.35	0.015
Matching Method	1	19797772	19797772	19797772	6.44	0.028
URL	1	232358	232358	232358	0.08	0.788
Matching Sim	1	6894093	6894093	6894093	2.24	0.162
Error	11	33810800	33810800	3073709		
Total	15	86393045				

S = 1753.20 R-Sq = 60.86% R-Sq(adj) = 46.63%



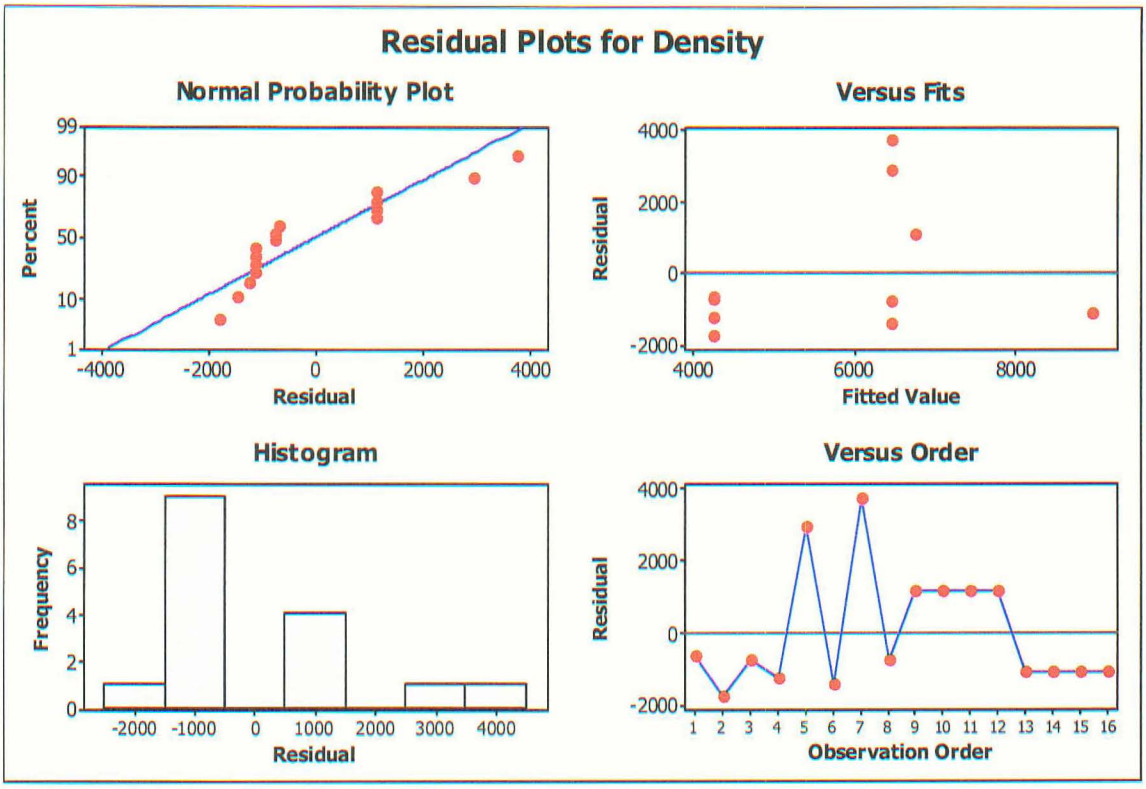
General Linear Model: Density versus Method, Matching Method

Factor	Type	Levels	Values
Method	fixed	2	0, 1
Matching Method	fixed	2	0, 1

Analysis of Variance for Density, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Method	1	25658023	25658023	25658023	8.15	0.014
Matching Method	1	19797772	19797772	19797772	6.29	0.026
Error	13	40937251	40937251	3149019		
Total	15	86393045				

S = 1774.55 R-Sq = 52.62% R-Sq(adj) = 45.33%



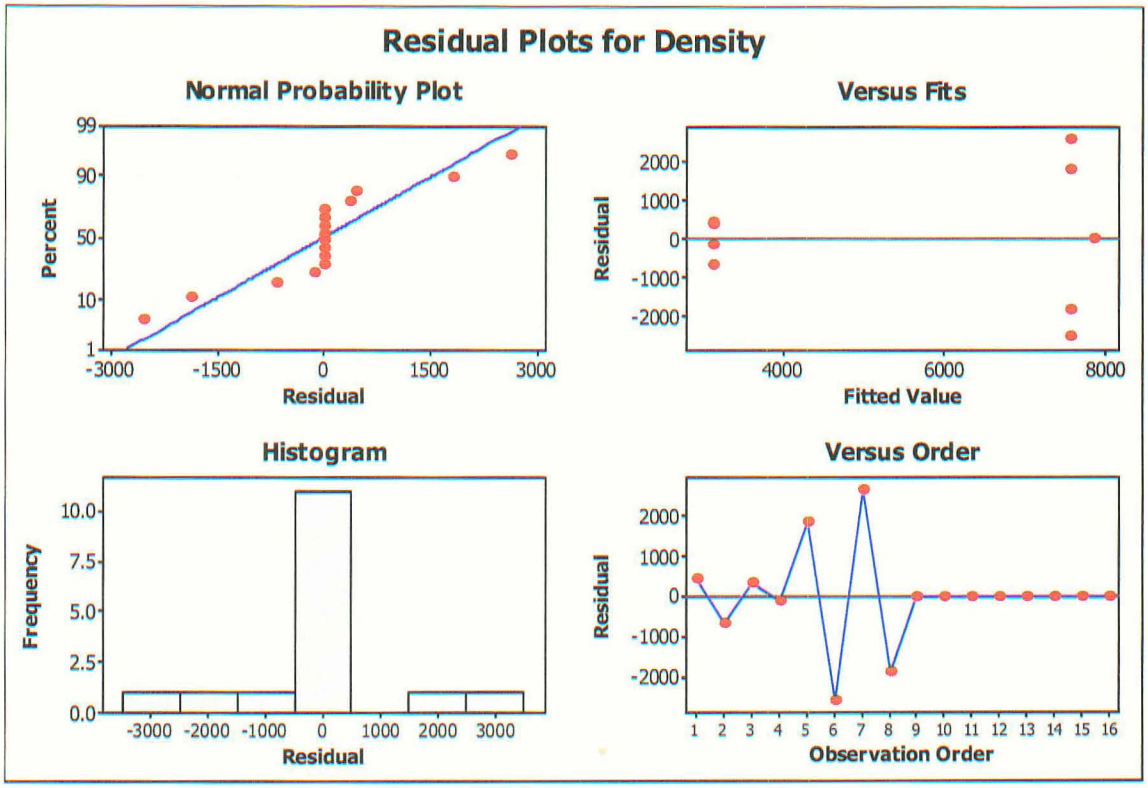
General Linear Model: Density versus Method, Matching Method

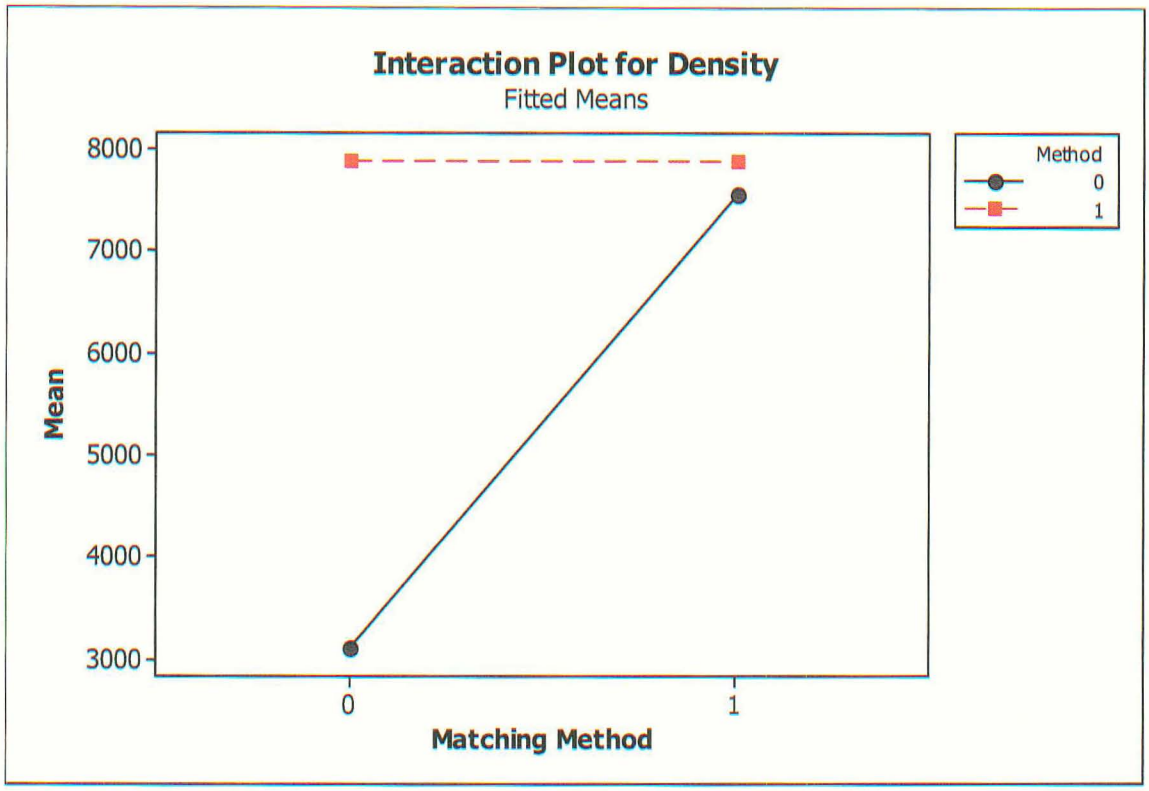
Factor	Type	Levels	Values
Method	fixed	2	0, 1
Matching Method	fixed	2	0, 1

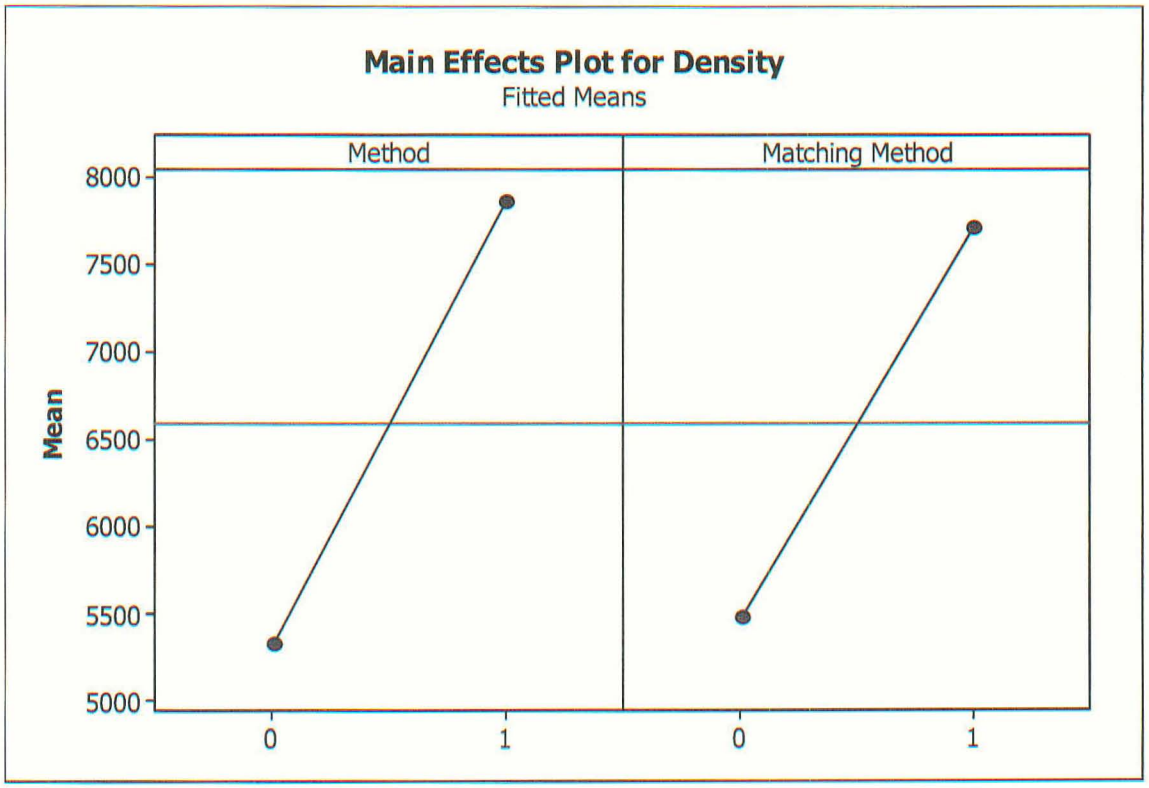
Analysis of Variance for Density, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Method	1	25658023	25658023	25658023	14.56	0.002
Matching Method	1	19797772	19797772	19797772	11.24	0.006
Method*Matching Method	1	19797772	19797772	19797772	11.24	0.006
Error	12	21139479	21139479	1761623		
Total	15	86393045				

S = 1327.26 R-Sq = 75.53% R-Sq(adj) = 69.41%




Interaction Plot for Density
Fitted Means
Method
0
1
Mean
8000
7000
6000
5000
4000
3000
0
1
Matching Method


Main Effects Plot for Density
Fitted Means
Method
Matching Method
Mean
8000
7500
7000
6500
6000
5500
5000
0
1
0
1

CURRICULUM VITAE

NAME: Basheer Hawwash

ADDRESS: 3012 Hikes Lane
Louisville, KY, 40220

DOB April 30, 1984

EDUCATION

B.S. Computer Information Systems
Jordan University of Science and Technology
Irbid, Jordan
2002-2006

M.S. Computer Science
University of Louisville
Louisville, KY, USA
2006-2008