TOWARD BETTER WEBSITE USAGE:
LEVERAGING DATA MINING TECHNIQUES AND ROUGH SET LEARNING TO
CONSTRUCT BETTER-TO-USE WEBSITES

A Dissertation
Presented to
The Graduate Faculty of The University of Akron

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Natheer Yousef Khasawneh
August, 2005
TOWARD BETTER WEBSITE USAGE:
LEVERAGING DATA MINING TECHNIQUES AND ROUGH SET LEARNING TO
CONSTRUCT BETTER-TO-USE WEBSITES

Natheer Yousef Khasawneh

Dissertation

Approved:

Advisor
Dr. John Durkin

Committee Member
Dr. John Welch

Committee Member
Dr. James Grover

Committee Member
Dr. Yueh-Jaw Lin

Committee Member
Dr. Yingcai Xiao

Accepted:

Department Chair
Dr. Jose De Abreu-Garcia

Dean of the College
Dr. George Haritos

Dean of the Graduate School
Dr. George Newkome

Date

Department Chair
Dr. Jose De Abreu-Garcia

Dean of the College
Dr. George Haritos

Dean of the Graduate School
Dr. George Newkome

Date
ABSTRACT

When users browse a website, they usually try to accomplish a certain task, such as finding information, buying products, registering for classes, and attending classes online. The interaction between the users and the website can give the web engineers insight into the most common user tasks performed on the website. They can learn how most users navigate the website to finish their tasks and what changes can be made to the website structure in order to make the completion of the common tasks easier and faster. Most web servers provide web interaction logs to track the interaction between the users and the website. But such logs are usually designed for debugging purposes and not for the analysis of the website. So there is a need for a deeper conceptual method to analyze the interaction log to reveal information that can be used for enhancing the website structure.

In this work, different data mining techniques, along with a rough set learning approach, are presented to enhance website usage. A new active-user-based user identification algorithm was applied to the interaction log to group together records that belong to the same user. The algorithm has a complexity running time of one order faster than other user identification algorithms. Sessions for identified users are found using an ontology-based session identification algorithm, which uses the website ontology in determining the sessions within website users. Different website sessions are then compared using a new Multidimensional Session Comparison Method (MSCM). MSCM
takes into consideration other session dimensions, such as pages visited, time spent on the pages and the session length. MSCM compares sessions more precisely than other well known session comparison methods, such as the Sequence Alignment Method (SAM), Multidimensional Sequence Alignment Method (MDSAM), and Path Feature Space. Using the comparison results from the MSCM, sessions are clustered by hierarchal and equivalence classes clustering algorithms. The clustering results are used by the rough set learning method and the centroid method to generate rules that are used for both predicting and describing sessions’ clusters. Rules generated using a rough set learning approach predict and describe clusters better than rules generated using centroid method. Each session cluster is considered one task and the cluster centroid is the navigation path for completing the task. So common tasks along with their navigation path are evaluated, suggestions are then made for the website engineer to enhance the website structure to better serve website users. This work shows how data mining techniques along with rough set learning methods can be used to enhance the website structure for better-to-use websites.
DEDICATION

To my parents...
ACKNOWLEDGEMENTS

All praises are due to ALLAH (GOD). Every good comes through HIM alone. So praises be to HIM.

My profound thanks to my advisor Dr. John Durkin for his support, confidence, and understanding. My deep appreciations to Dr. C.-C. Chan for his constant support and insightful guidance and to Dr. Tom Xiao for the good time I spent with him in the ODOT project that was very helpful in my research. I want to thank Dr. John Welch for his proof readings and the time he spent with me teaching in the "Tools Lab." Dr. James Grover and Dr. Y.-J. Lin also gave me invaluable support throughout my research. My special thanks to the staff in the computer center at the University of Akron for providing the data for this research. My thanks also go to the faculty and the staff of the Department of the Electrical and Computer Engineering for their support.

My heartfelt thanks to my brothers in Akron Majsid, Abdul Kareem, Abdul Raheem, Yahya, Hussien, Masoud, Musa and Abdel Ghanee, for their prayers and support. To my dear friends in USA, including, Qasem, Luay, Qais, Ahmad, Mohammad, Hussein, Huthaifa, Faisal, Sami, Samer, and Majed a special thanks for the happy time we spent together.

My friends and family in Jordan, including my mother, Mrs. Fairouze Khasaswneh, my father, Mr. Yousef Khasawneh, my sisters Fatemah, Hala, and Dr. Maha, my brothers Dr. Basheer and Dr. Mohammad and their families, have been my strongest support
system. This project surely would not have been accomplished without their love, care and DU'A (prayers).
TABLE OF CONTENTS

LIST OF TABLES………………………………………………………………… xiii
LIST OF FIGURES………………………………………………………………… xv

CHAPTER

I. INTRODUCTION…………………………………………………………… 1
   1.1 Motivation ……………………………………………………………… 2
   1.2 Previous work ………………………………………………………… 2
   1.3 Proposed WUM system architecture…………………………… 3
   1.4 Main contributions ………………………………………………… 5
   1.5 Research objective………………………………………………… 6
   1.6 Structure of the dissertation……………………………………… 6

II. WEB LOG DATA PREPROCESSING FOR WEB USAGE MINING … 7
   2.1 Introduction…………………………………………………………… 7
   2.2 Previous work………………………………………………………… 9
   2.3 Data preprocessing architecture………………………………… 10
      2.3.1 Data cleaning………………………………………………… 10
      2.3.2 User identification…………………………………………… 11
         2.3.2.1 User identification problem statement………………… 12
         2.3.2.2 A trivial user identification algorithm………………… 12
5.1 Introduction................................................................................. 79
5.2 Data preparation module......................................................... 80
5.3 Session identification module.................................................. 81
5.4 Clustering process module....................................................... 84
5.5 Results presentation and evaluation module............................. 88
5.6 Summary.................................................................................. 90

VI. SUMMARY AND CONCLUSIONS..................................................... 92
REFERENCES.................................................................................. 96
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>20</td>
</tr>
<tr>
<td>2.2</td>
<td>21</td>
</tr>
<tr>
<td>2.3</td>
<td>21</td>
</tr>
<tr>
<td>2.4</td>
<td>28</td>
</tr>
<tr>
<td>2.5</td>
<td>28</td>
</tr>
<tr>
<td>2.6</td>
<td>29</td>
</tr>
<tr>
<td>3.1</td>
<td>36</td>
</tr>
<tr>
<td>3.2</td>
<td>37</td>
</tr>
<tr>
<td>3.3</td>
<td>41</td>
</tr>
<tr>
<td>3.4</td>
<td>42</td>
</tr>
<tr>
<td>3.5</td>
<td>43</td>
</tr>
<tr>
<td>3.6</td>
<td>44</td>
</tr>
<tr>
<td>4.1</td>
<td>47</td>
</tr>
<tr>
<td>4.2</td>
<td>58</td>
</tr>
<tr>
<td>4.3</td>
<td>58</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.4</td>
<td>Example of a rule generated by a web sessions' classifier</td>
</tr>
<tr>
<td>4.5</td>
<td>Decision produced by the clustering algorithm</td>
</tr>
<tr>
<td>4.6</td>
<td>Certain rules learned from 4.5 using BLEM2</td>
</tr>
<tr>
<td>4.7</td>
<td>Inference engine testing examples</td>
</tr>
<tr>
<td>4.8</td>
<td>Inference engine results along with results from cluster</td>
</tr>
<tr>
<td>6.1</td>
<td>Mathematical models for three website parameters</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Proposed WUM system architecture</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>Data preprocessing architecture</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>Formal user identification problem statement</td>
<td>12</td>
</tr>
<tr>
<td>2.3</td>
<td>Trivial user identification algorithm</td>
<td>13</td>
</tr>
<tr>
<td>2.4</td>
<td>Active user-based user identification algorithm</td>
<td>15</td>
</tr>
<tr>
<td>2.5</td>
<td>Ontology-based session identification algorithm</td>
<td>18</td>
</tr>
<tr>
<td>2.6</td>
<td>Monthly record counts recorded in the web log</td>
<td>20</td>
</tr>
<tr>
<td>2.7</td>
<td>Active user-based user identification script</td>
<td>23</td>
</tr>
<tr>
<td>2.8</td>
<td>Histogram for sessions’ lengths after session identification</td>
<td>24</td>
</tr>
<tr>
<td>2.9</td>
<td>Histogram for the sessions’ lengths before filtering</td>
<td>25</td>
</tr>
<tr>
<td>2.10</td>
<td>Probability of the number of records per user</td>
<td>27</td>
</tr>
<tr>
<td>2.11</td>
<td>Probability of inactive user time in seconds</td>
<td>28</td>
</tr>
<tr>
<td>2.12</td>
<td>Probability of recorded records per second</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>Web usage classification and prediction workflow</td>
<td>48</td>
</tr>
<tr>
<td>4.2</td>
<td>Two well separated clusters with intermediate chain</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>Hierarchal clustering algorithm</td>
<td>55</td>
</tr>
<tr>
<td>4.4</td>
<td>Equivalence classes clustering algorithm</td>
<td>57</td>
</tr>
</tbody>
</table>
4.5 Percentage of the number of the clusters from the initial number of clusters for a specific session length at different iterations using the average linkage method………………………………………………………… 60

4.6 Percentage of the number of the clusters from the initial number of clusters for a specific session length at different iterations using the Ward’s method………………………………………………………… 60

4.7 Holdout classifier accuracy estimator…………………………… 67

4.8 Percentage of the number of the clusters from the initial number of clusters for different session length groups at different iterations…… 70

4.9 Average accuracy for different session lengths at the 100% number of clusters using examples from the hierarchal clustering algorithm……… 71

4.10 Average accuracy for different session lengths at the 15.69% number of clusters using examples from the hierarchal clustering algorithm……… 72

4.11 Average accuracy for different session lengths at the 15.69% number of clusters using examples from the equivalence classes clustering algorithm………………………………………………………………………. 73

4.12 Cluster description length for different session lengths using different classifiers………………………………………………………………………………………………………………………………………………………………………………………. 74

4.13 Seven most common tasks performed on the website…………….. 75

4.14 Sequence length distribution for “Class Search Detail”…………….. 76

5.1 Data flow diagram for the web usage mining system………………….. 80

5.2 Entity relation model for data preparation………………………….. 81

5.3 Use case diagram for session identification………………………….. 82

5.4 Session identification module user interface………………………….. 84

5.5 Use case diagram for clustering process module……………………….. 85

5.6 UML diagram for the clustering process……………………………….. 85

5.7 Sequence diagram for generating dissimilarity matrix…………………… 86

5.8 Sequence diagram finding clusters……………………………………… 86
5.9 Clustering module user interface .............................................. 88
5.10 Dataflow diagram for the results presentation and evaluation module..... 90
CHAPTER I

INTRODUCTION

The World Wide Web has greatly impacted every aspect of our societies and our lives. This ranges from information dissemination to communication, and from e-commerce to process management. By browsing through a website, users complete different tasks, such as buying products, registering for classes, and attending classes online. Web Usage Mining (WUM), a new field that analyzes the navigation process, has emerged in recent years. WUM is defined as applying data mining techniques to log interactions between users and a website [1]. Analysis of an interaction log file can provide useful information that helps a website engineer in enhancing the website structure in a way that will make the website usage easier and faster in the future.

In this dissertation, we are interested in the clustering of web users’ sessions in the context of web applications, such as registration web-based systems, distance-education web-based systems, e-commerce sites, and any other web based applications. Clustering web users’ sessions is to group users with similar navigation behaviors together. Our goal is to use the clustering results to identify dominant browsing behaviors, evaluate a website structure and predict future users’ browsing behaviors to better assist users in their future browsing experiences.
1.1 Motivation

In the process of designing a web application it is hard to predict how users will use a website in completing different tasks. Usually, web designers have the choice to make a certain task easier to complete than other tasks by constructing the website structure in a certain way. After publishing the website online and having users interact with it for a while, it becomes the time to review certain decisions concerning the website structure. Such decisions can be made by analyzing the interaction log between users and the website. A deep conceptual analysis of the interaction log is required to understand what the most common tasks done over the website are, how the majority of users navigate the website to achieve such common tasks, and what changes can be made to the website structure to make the completion of the common tasks easier and faster. For example, if we have a registration website—where users can do different tasks, such as check grades, add classes, drop classes, and pay tuition fees—there is a need for a system to determine what the most common tasks are, and how easily they can be achieved by users. So, if we find that, at a certain point in time, the grade checking process is the most common task, and it takes a long time for users to finish this task, the website engineer shall be advised to enhance the website structure to make this task easier and faster.

1.2 Previous work

Available commercial web usage mining systems, such as Surfaid [2], Net Tracker [3], and WebTrends [4], give statistical information about the website, such as the average usage hits, geographical distribution of users, and the most frequent page hit. These are considered to be statistically significant results rather than conceptual results. For example, if we conclude that during a certain time a given number of users hit a
website, this gives no insight into the hidden usage patterns. Other published work on web usage mining used different data mining techniques such as association rules [5], clustering and classification. In our research, we focus on researches that use clustering and classification of data mining techniques. For example, Fu et al. [6] used the BIRCH [7] clustering algorithm to cluster users’ sessions. However, they did not discuss how the closeness between different sessions was defined, and they did not show how they chose the maximum difference allowed between sessions in the same cluster. Foss et al. [8] presented a novel clustering algorithm that clusters users’ sessions. Their clustering algorithm did not require any input parameter from the users, such as the final number of clusters, or the maximum difference allowed between sessions in the same cluster. The way they measure similarity did not consider the order of the pages. For example, they considered a session consisting of pages, say A, B, and C, identical to a session consisting of pages, say A, B, C, and D, or any other number of pages that contains the pages A, B, and C.

1.3 Proposed WUM system architecture

As shown in Figure 1.1, the proposed WUM system is divided into four phases: preprocessing, dissimilarity measure, clustering analysis, and results incorporation and evaluation. In the preprocessing phase, the raw web logs are filtered from unrelated web requests, the records that belong to the same user are then grouped together in one set. In the dissimilarity measure phase, the users’ records are divided into one or more sessions and a dissimilarity matrix, which reflects the dissimilarity between different sessions, is constructed. In the clustering analysis phase, sessions with similar browsing behaviors are grouped together. In the last phase—the results incorporation and evaluation phase—
clustering results are incorporated to predict future users’ classes, and present suggestions to enhance the website structure to adequately better serve website users in their future visits.

Figure 1.1 Proposed WUM system architecture
1.4 Main contributions

In each WUM phase presented in Section 1.3, we have one or more new contributions to the WUM field.

In the preprocessing phase, we present a fast active user-based user identification algorithm with time complexity $O(n)$. For the session identification phase, we present an ontology-based session identification algorithm that utilizes the website structure and its functionalities in identifying different sessions. In addition, we present extra cleaning steps, such as removing housekeeping pages, removing redundant pages, and grouping sessions with similar session lengths. We also present three mathematical models for the parameters on which our user-identification algorithm depends.

In the dissimilarity measure phase, we present a new Multidimensional-Sessions Comparison-Method (MSCM) using dynamic programming. Our method takes into consideration different session dimensions, such as the page list, the time spent on each page, and the length of the session. This is in contrast to other algorithms that treat sessions as sets of visited pages within a time period and do not consider the sequence of the click-stream visitation or the session length.

In the web sessions clustering analysis phase, we present two clustering algorithms: a hierarchal clustering algorithm and an equivalence classes clustering algorithm. The equivalence classes clustering algorithm does not depend on the seed starting point of the clustering process. We also present a new method to determine the clustering parameter that in turn determines where the clustering algorithm should stop.

In the results incorporation and presentation phase, we present a rough set approach in predicting the future classes and we present the results in a way that can be
incorporated in the website server for predicting future users’ classes. We also present an evaluation process that evaluates the accuracy of the predicted classes. Finally, we show how results can be incorporated to enhance the website structure to better serve future website users.

1.5 Research objective

Our main objective in this research is to present a WUM system that uses clustering algorithms along with a rough set learning approach. This improved WUM system presents a deep conceptual understanding of the usage behavior for a website, can be used by the website engineer to evaluate and enhance the website structure, and to predict “what the user was trying to do” to better assist users in their future browsing experiences. This should lead to websites that are easier and more convenient for users to navigate.

1.6 Structure of the dissertation

The rest of the dissertation is organized as follows. In Chapter 2, we present the data preprocessing phase. In Chapter 3, we present the MSCM method. In Chapter 4, we present the four main steps incorporated in the WUM system: the clustering algorithms, the clustering results presentation, evaluating the results and incorporating the results to improve the website structure. In Chapter 5, we present an overview of the system implementation. In Chapter 6, we present conclusions drawn from the research and recommendations for future work.
Web usage mining is “the application of data mining techniques to large Web data repositories in order to extract usage patterns” [1]. Web log files contain data that need some cleaning since their formats were meant for debugging purposes only [9]. In this chapter, we present new techniques for preprocessing web log data and for identifying unique users and sessions from the data. We present a fast active user-based user identification algorithm with time complexity $O(n)$. The algorithm uses both an IP address and a finite users’ inactive time to identify different users in the web log. For the session identification, we present an ontology-based session identification method that utilizes the website structure and functionalities to identify different sessions. In addition, we present extra cleaning steps such as removing housekeeping pages, removing redundant pages, and grouping sessions with similar session lengths. Finally, we present three mathematical models for the website parameters on which our active user-based user identification algorithm depends.

2.1 Introduction

Web usage mining is “the application of data mining techniques to large Web data repositories in order to extract usage patterns” [1]. Data mining techniques—such as association rule mining, sequential patterns or clustering analysis—cannot be applied
directly to raw web-log-files data, since the format of this data was designed for debugging purposes [9]. Five preprocessing steps have been identified [10]:

1. **Data cleaning**: This step removes irrelevant data, such as log records for images, scripts, help files, and cascade style sheets. Only data that is relevant to the mining process is kept.

2. **User identification**: This consists of grouping together records for a same user. Log records are recorded in a sequential manner as they are coming from different users (i.e., records for a specific user may not necessarily be in consecutive order since they could be separated by records from other users).

3. **Session identification**: This step divides the pages access of each user into individual sessions.

4. **Path completions**: This step determines whether there are important accesses that are not recorded in the access log due to caching on several levels.

5. **Formatting**: This step formats the data to be readable by data mining systems.

In this chapter, we present a detailed data processing architecture that includes data cleaning, user identification, session identification and data filtering. Our main contributions in this chapter include a user identification algorithm that is running in a time complexity of $O(n)$, an ontology-based session identification algorithm, and three mathematical models for the website parameters on which our user identification algorithm depends.

The rest of the chapter is organized as follows. In Section 2, we survey the previous work on web usage mining preprocessing techniques. In Section 3, we present our data preprocessing architecture along with a detailed description for each step. In Section 4,
we present experimental results. Section 5 presents the mathematical models for different website parameters. In Section 6, we present the summary.

2.2 Previous work

Previous work on preprocessing web logs emphasized on the caching problem since caching produces incomplete web logs. One solution to this is to collect the data on the client side. For example, Shahabi [11] collected almost all the user interactions with the browser. Fenstermacher and Ginsburg [12] went beyond the browser interaction and recorded the interaction between the user and some other applications. Catledge and Pitkow [13] presented another system that collected data on the client side. The use of these methods imposes the issue of security. Moreover, most of the proposed methods require special browsers and setups.

Other works like [14] and [15] assumed that the filtered web server log is a good representation for web usage meaning and that there is no need for any heuristic methods to complete the sequences such as path completion process [10].

In their task of the preprocessing step [16], Yan et al. converted the information in user access logs into a vector representation. The vector representation combines the page access along with the amount of interest a user shows in a page, which was calculated by counting the number of times the page was accessed.

Path completion [10] identifies missing records in users’ sessions using a heuristic method, which is based on the web structure. Transactions were identified either by reference length, which is based on the time spent on the page, or by maximal forward reference, which is based on the first backward action (hitting the back button on the browser) after a series of forward actions (normal forward navigation). Other heuristic
methods like time-oriented heuristics [17] and navigation-oriented heuristics [18] were used to identify different sessions.

It can be concluded that previous work has not identified a specific algorithm for user identification; rather, they assumed that users’ records are readily available in the website log.

2.3 Data preprocessing architecture

As shown in Figure 2.1, we identified four steps in the data preprocessing phase: data cleaning, user identification, session identification and data filtering. The following subsections provide details on each step.

![Data preprocessing architecture](image)

**Figure 2.1 Data preprocessing architecture**

2.3.1 Data cleaning

Web logs are designed for debugging purposes in that the web accesses are recorded in the order they arrive [9]. Due to the connectionless nature of the HTTP (i.e., each request is handled in a separate connection), web log records for a single user do not necessarily appear contiguously since they could be interleaved with records from other users. Thus, for each page component—such as an image, a cascading style sheet file, an HTML file, scripting file, or a Java script—a separate record is recorded in the web log file. Usually, each record in the web-log file has the following standard format [19]:

```
Records from Web Log → Data Cleaning → User Identification

Data Mining Technique ← Data Filtering ← Session Identification
```

10
• *Remotehost* which is the remote hostname or its IP address;

• *Logname* which is the remote logname of the user;

• *Date*, which is the date and time of the request;

• *Request*, which is the exact request line as it came from the user;

• *Status*, which is the HTTP status code returned to the client; and

• *Byte*, which is the content-length of the document transferred.

Usually, for web mining purposes the only interesting elements are the HTML pages and the scripting pages—such as JSP, ASP or PHP pages—unless other file types are playing a navigation role in the web application and they are part of the web structure. In the cleaning phase, the file types that are related to the navigation structure are kept and other files are eliminated. The *status* field in the web log can be used to keep the successfully fulfilled requests and to delete the unsuccessful requests. Finally, the mining process can be limited to a certain time or date so that web traffic during such time and date will be only considered.

2.3.2 User identification

A user is defined as a unique client to the server during a specific period of time. The relationship between users and web log records is one to many (i.e., each user is identified by one or more records). Users are identified based on two assumptions:

1. Each user has a unique IP address while browsing the website. The same IP address can be assigned to other users after the user finishes browsing.

2. The user may stay in an inactive state for a finite time after which it is assumed that the user left the website.
Next, we formally present the problem statement of the user identification, and then we present two different algorithms for user identification: a trivial algorithm and our new active user-based one.

2.3.2.1 User identification problem statement

Figure 2.2, shows a formal description of the user identification problem statement. As stated earlier, the user identification algorithm must identify the user’s record based on the assumption that all user’s records have the same IP address and a finite inactive browsing time, \( \beta \).

<table>
<thead>
<tr>
<th>Given web log record ( R = \langle r_1, \ldots, r_k \rangle ), where ( k &gt; 0 ) and ( k ) is the total number of records in the web log database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \forall r \in R, r ) is defined as</td>
</tr>
<tr>
<td>( \langle \text{date_time}, c_{-ip}, s_{-ip}, s_{-port}, cs_{-method}, url, url_{query}, status, s_{-agent} \rangle )</td>
</tr>
<tr>
<td>Find users ( U = \langle u_1, \ldots, u_j \rangle ) ( \forall u \in U, u ) is defined as</td>
</tr>
<tr>
<td>( u = \langle c_{-ip}, last_{-date}_{-time}, {r_s, \ldots, r_e} \rangle )</td>
</tr>
<tr>
<td>( \forall r \in u, r.c_{-ip} = c_{-ip} ) and ( r.date_{-time} \leq last_{-date}_{-time} + \beta ) at the time record ( r ) is added to the user ( u )</td>
</tr>
<tr>
<td>where:</td>
</tr>
<tr>
<td>( c_{-ip} ) is the user’s ( ip ) address, ( last_{-date}_{-time} ) is the date and time when the user accessed the last record, ( \beta ) is the maximum user’s idle time, ( r_s ) is the first record the user accessed in a single visit to the website, and ( r_e ) is the last record the user accessed in a single visit to the website.</td>
</tr>
</tbody>
</table>

Figure 2.2 Formal user identification problem statement

2.3.2.2 A trivial user identification algorithm

Figure 2.3, shows the trivial user identification algorithm. The figure shows that the algorithm has two loops: an outer loop; and inner. The outer loop has time complexity of \( n \), where \( n \) is the size of the total records. The inner loop has time complexity of \( i \), where \( i \) is the total number of current users. In the worst case, each user has one record which
leads to two loops, the outer and the inner, each of size $n$. Hence, the overall time
complexity of the algorithm is $O(n \times n) = O(n^2)$.

```
Assumption:
n number of records in the web log

Define
R : Website records
U : Users' records
R(i) : i_{th} Weblog record

Initialize U = \emptyset
u_{first}.c_ip = R(1).c_ip
u_{first}.last_date_time = R(1).date_time
u_{first}.r = R(1)
U = U \cup \{u_{first}\}

for each record r \in R
  for each record u \in U
    if(r.c_ip = u.c_ip AND r.date_date \leq u.last_date_time + \beta)
      u.r = u.r \cup r
    else
      u_new.c_ip = r.c_ip
      u_new.last_date_time = r.date_time
      u_new.r = r
      U = U \cup \{u_{new}\}
  endif
endfor
endfor
```

Figure 2.3 Trivial user identification algorithm

2.3.2.3 The active user-based user identification algorithm

The algorithm shown in Figure 2.4 is a modified version of the algorithm described
in Section 2.3.2.2. We limited the inner loop search to the active users only. Active users
are defined as users who did not exceed the maximum inactive time, and hence they are considered to be still browsing the website and more records are likely to be added to their navigation records.

Time complexity analysis shows that there are two loops: outer and inner loops. The outer loop has time complexity of $n$, where $n$ is the size of the total records. The inner loop has time complexity of $i$ where $i$ is the total number of active users. According to the assumptions given at the beginning of the algorithm, the maximum number of active records cannot exceed $(m \cdot k \cdot t)$ in the worst case, where $m$ is the number of records per user, $k$ is the rate of the recorded records in the web log, and $t$ is the inactive browsing time. So, the algorithm breaks down to two loops: the outer loop with the size of $i$, and the inner loop with the size of $(m \cdot k \cdot t)$, which is constant. Therefore, the overall complexity of the algorithm becomes $O((m \cdot k \cdot t) \cdot n) = O((cons.) \cdot n) = O(n)$. 
Assumption
\( n \) records in the web log
\( m \) number of records per user
\( k \) records/second recorded in the web log
\( t \) inactive time in seconds for user
\( \beta \) maximum inactive time in seconds for the user

Define
\( U_A : \) Active users; users who still browsing
\( U_I : \) Idle users; users who stopped browsing

Initialize \( U_A = \) first record in \( R \)

Initialize \( U_I = \emptyset \)

for each record \( r \in R \) do [starting at the 2nd record]
  for each record \( u \in U_A \) do
    if \( (r.\text{date}\_\text{time} > u.\text{last}\_\text{date}\_\text{time} + \beta) \)
      \( U_A = U_A \setminus \{u\} \) //Remove record from active users list
      \( U_I = U_I \cup \{u\} \) //Add record to the idle users list
    else if \( (r.\text{c}\_\text{ip} = u.\text{c}\_\text{ip} \text{ AND } r.\text{date}\_\text{time} \leq u.\text{last}\_\text{date}\_\text{time} + \beta) \)
      \( u.r = u.r \cup r \)
      if \( (r.\text{date}\_\text{time} > u.\text{last}\_\text{date}\_\text{time}) \)
        \( u.\text{last}\_\text{date}\_\text{time} = r.\text{date}\_\text{time} \)
    endif
  else
    \( u_{new}.\text{c}\_\text{ip} = r.\text{c}\_\text{ip} \)
    \( u_{new}.\text{last}\_\text{date}\_\text{time} = r.\text{date}\_\text{time} \)
    \( u_{new}.r = r \)
    \( U_A = U_A \cup \{u_{new}\} \) //Add new record to the active users list
  endif
  endfor

\( U_I = U_I \cup U_A \) //Add the remaining records to the idle users list

Figure 2.4 Active user-based user identification algorithm
2.3.3 Ontology-based session identification

A session is defined as the stream of mouse clicks whereby a user is trying to perform a specific task. In our research, we compare different task specific browsing behavior. For example, assume two users, A and B, performed the following tasks.

User A searched for classes and checked his grade; whereas, user B paid his tuition fees and searched for classes.

The user identification process identifies users A and B as two separate users with totally different behaviors. However, if we divide each user into different sessions, user A will have two sessions: searching for classes and checking for grades; user B will also have two sessions: searching for classes and paying tuition. This shows that users A and B are partially similar in searching for classes rather than being totally different if using only user identification process.

We identify different sessions in a single user visit using the website ontology. We also assume that the website ontology is already available through methods of retrieving website ontology like the ones in [20-22].

The website ontology is defined as $W = (P, L, F)$ where:

$P$: Website pages,

$L$: Website links,

$F$: Website functionalities,

$P = \langle p_1, \ldots, p_k \rangle$ where $k$ is the number of pages in the website.
Links $L$ is the group of links for the web application. Each link $l = (p_s, p_d)$ is defined by two pages: the source page ($p_s$) where a link starts from, and the destination page ($p_d$) where links ends.

Define web functionalities $F$ as $F = \langle f_0, f_1, \ldots, f_{n-1} \rangle$, where $\forall f \in F, f = \langle p_s, \ldots, p_e \rangle$. Each web functionality, $f$, consists of at least two pages: a start page and an end page. There can be zero or more pages between the start and end pages. The session identification algorithm will divide the users identified in the Section 2.3.2 into different sessions using the website functionalities. From the website functionalities, we can identify pages that are considered as the breaking points for the session, such as the sign-in or the sign-out pages.

Figure 2.5 shows the ontology-based session identification algorithm, where $B$ is the set of the breaking pages. The algorithm splits each user into one or more sessions and returns a final list of sessions $S$. The time complexity analysis of the algorithm shows two loops: an inner and an outer loop. The inner loop depends on the number of records per user, $m$, and the outer loop depends on the total number of users, $j$. It can be easily concluded that $m \cdot j = n$, where $n$ is the total number of records. So, the overall time complexity of the algorithm is $O(n)$. 
2.3.4 Data filtering

After we identify different users’ sessions, filtering is done based on removing the housekeeping pages. The housekeeping pages are the pages that are necessary for the web application to run properly. They are not called directly by the user; rather, they are called internally by the requested page. These pages are identified by the website engineer and can be found using the website ontology. Removing the housekeeping pages can result in redundant pages which can be misleading to the sequence comparison method, which we present in Chapter 3. To illustrate this, consider the following two sequences:

Sequence 1: $p_2 \rightarrow p_1 \rightarrow p_3 \rightarrow p_4 \rightarrow p_5 \rightarrow p_1 \rightarrow p_5 \rightarrow p_6$,

Sequence 2: $p_2 \rightarrow p_7 \rightarrow p_8 \rightarrow p_4 \rightarrow p_5 \rightarrow p_6$

It can be seen that these two sequences are totally different. However, assuming that pages $p_1, p_3, p_7,$ and $p_8$ are housekeeping pages and applying the following two step filtering process, sequence 1 and 2 reduces to:

Step 1: Removing the housekeeping pages
Sequence 1 becomes $p_2 \rightarrow p_4 \rightarrow p_5 \rightarrow p_6$

Sequence 2 becomes $p_2 \rightarrow p_4 \rightarrow p_5 \rightarrow p_6$

Step 2: Removing the redundant pages

Sequence 1 becomes $p_2 \rightarrow p_4 \rightarrow p_5 \rightarrow p_6$

Sequence 2 becomes $p_2 \rightarrow p_4 \rightarrow p_5 \rightarrow p_6$

The outcome of the two-steps filtering process shows how sequences that might look strikingly different they are actually similar.

2.4 Experimental results

For experimental results, we used data obtained from the University of Akron registration website log files during the period from October 2003 to September 2004. In this section, we present the results from each preprocessing step mentioned in the previous section.

2.4.1 Data overview

The total number of records recorded in the web server for the time period mentioned was 28,294,229 records. Each record in the web log file represents a page request processed by the web server. Figure 2.6 shows the traffic volume on the web server over the selected time period. The figure shows high web traffic during the months when there was a major activity such as beginning registration, release of final grades, or beginning of a semester. For example, it is clear from the figure that there was a high traffic volume in January, which is the time just after the release of the final grades for fall semester and just before the spring semester beginning.
2.4.2 Data selection process

For experimental purposes, we selected the data records for the days in which there was major activity on the web server. Table 2.1 shows the selected dates along with the major activity. The total number of records for the selected dates was 1,582,292, which represents 5.6% of the total records.

Table 2.1 Selected dates for experimental results along with their major activity

<table>
<thead>
<tr>
<th>Date</th>
<th>Major Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday, December 15, 2003</td>
<td>Teachers upload students grades</td>
</tr>
<tr>
<td>Tuesday, December 16, 2003</td>
<td>Final grades due for fall semester 2003</td>
</tr>
<tr>
<td>Monday, May 10, 2004</td>
<td>Teachers upload students grades</td>
</tr>
<tr>
<td>Tuesday, May 11, 2004</td>
<td>Final grades due for spring semester 2004</td>
</tr>
<tr>
<td>Friday, February 20, 2004</td>
<td>Summer semester registrations begin</td>
</tr>
<tr>
<td>Friday, October 24, 2003</td>
<td>Spring semester registrations begin</td>
</tr>
<tr>
<td>Friday, April 02, 2004</td>
<td>Fall semester registration begin</td>
</tr>
</tbody>
</table>

2.4.3 Data cleaning results

Table 2.2 shows the percentage of different file types in the selected data set. Since we are interested in the scripting files that imply a direct request by the user, we kept the ASP and HTML file types and we removed other file types.
Table 2.2 The percentage of different file types in the selected data set

<table>
<thead>
<tr>
<th>File Type</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML</td>
<td>33307</td>
<td>2.10%</td>
</tr>
<tr>
<td>DLL</td>
<td>186</td>
<td>0.01%</td>
</tr>
<tr>
<td>No extension</td>
<td>2148</td>
<td>0.14%</td>
</tr>
<tr>
<td>PHP</td>
<td>2</td>
<td>0.00%</td>
</tr>
<tr>
<td>HTM</td>
<td>6938</td>
<td>0.44%</td>
</tr>
<tr>
<td>TXT</td>
<td>68</td>
<td>0.00%</td>
</tr>
<tr>
<td>ICO</td>
<td>645</td>
<td>0.04%</td>
</tr>
<tr>
<td>JPG</td>
<td>1510</td>
<td>0.10%</td>
</tr>
<tr>
<td>ASP</td>
<td>1537485</td>
<td>97.17%</td>
</tr>
<tr>
<td>JS</td>
<td>1</td>
<td>0.00%</td>
</tr>
<tr>
<td>XML</td>
<td>2</td>
<td>0.00%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1582292</strong></td>
<td><strong>100.00%</strong></td>
</tr>
</tbody>
</table>

The second cleaning step was to remove the uncompleted requests. This can be tracked using the status code in the log file described in Section 2.3.1. Table 2.3 shows the status code, the code description, the total number of records, and the percentage of records with specific status code. We kept the records with an HTML status code of *Ok* and removed the other records, so we are left with 76% of the total records.

Table 2.3 Requests status for the records in the web record

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Code Description</th>
<th>Page Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>206</td>
<td>Unknown</td>
<td>20</td>
<td>0.00%</td>
</tr>
<tr>
<td>207</td>
<td>Unknown</td>
<td>3</td>
<td>0.00%</td>
</tr>
<tr>
<td>304</td>
<td>No Change</td>
<td>17826</td>
<td>1.13%</td>
</tr>
<tr>
<td>302</td>
<td>Not Found</td>
<td>322728</td>
<td>20.40%</td>
</tr>
<tr>
<td>400</td>
<td>Bad request</td>
<td>74</td>
<td>0.00%</td>
</tr>
<tr>
<td>200</td>
<td>Ok</td>
<td>1206982</td>
<td>76.28%</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>17491</td>
<td>1.11%</td>
</tr>
<tr>
<td>404</td>
<td>Matching not found</td>
<td>1536</td>
<td>0.10%</td>
</tr>
<tr>
<td>501</td>
<td>Facility not supported</td>
<td>10</td>
<td>0.00%</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected condition</td>
<td>15622</td>
<td>0.99%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>1582292</strong></td>
<td><strong>100.00%</strong></td>
</tr>
</tbody>
</table>
2.4.4 User identification results

We loaded the selected data into a single table in an SQL database. Out of the six fields described in Section 2.3.1, we selected three fields (Remotehost, Date, and URL). Then, we ran the session identification script shown in Figure 2.7, which is based on the algorithm described in Section 2.3.2. To show the effectiveness of the active user based algorithm we ran both the active user based and the trivial user identification algorithm on the same records and we repeated the experiment using different web log sizes. The active user based algorithm shows much better performance over the other algorithm even for small web log sizes. For example, for 100 web log records, the trivial algorithm took 527 seconds to identify the users’ sequences, while the active user-based algorithm took 8 seconds. For the full log size (1,582,292 records), the trivial algorithm ran for about two days and was aborted by the operating system apparently because of the memory build up without giving any result; whereas the active user-based algorithm took only three hours and 33 minutes to yield the results.
DECLARE @RecordId int
DECLARE @Date DateTime
DECLARE @IPAddress varchar(255)
DECLARE @FoundUser_id int
DECLARE @NewUserId int

DECLARE UserCursor CURSOR FOR
SELECT id, date, c_ip FROM guest.weblog order by date
OPEN UserCursor
FETCH NEXT FROM UserCursor INTO @RecordId, @Date, @IPAddress
WHILE @@FETCH_STATUS = 0
BEGIN
--Delete old (more than 30 minutes old) records from active users
    DELETE FROM guest.active_users WHERE DATEADD(minute, 30, date) < @Date
--See if there is active users with the same ip iddress
    SET @FoundUser_id = (select top 1 user_id from guest.open_users where c_ip = @IPAddress)
    IF LEN(@FoundUser_id) > 0
        BEGIN
            --if yes, update the max time
            UPDATE guest.active_users SET date = @date where c_ip = @IPAddress AND user_id = @FoundUser_id
            --and insert new value into the users table,
            INSERT INTO guest.users(user_id, id) VALUES(@FoundUser_id, @RecordId)
        END
    ELSE
        BEGIN
            --if NO, insert new item in the open user table,
            --get the new user id and then ...
            INSERT INTO guest.active_users(c_ip, date) VALUES (@IPAddress, @Date)
            SET @NewUserId = @@IDENTITY
            -- Insert in the users table as well
            INSERT INTO guest.users(user_id, id) VALUES(@NewUserId, @RecordId)
        END
    END
FETCH NEXT FROM UserCursor INTO @RecordId, @Date, @IPAddress
END
CLOSE UserCursor
DEALLOCATE UserCursor

Figure 2.7 Active user-based user identification script
2.4.5 Session identification and data filtering results

Figure 2.8 shows the histogram for sessions’ lengths after session identification. The session identification was done based on two breaking pages *sign in* and *sign out*.

![Histogram for sessions’ lengths after session identification](image)

Figure 2.8 Histogram for sessions’ lengths after session identification

Grouping the sessions according to their length is important since some learning algorithms require fixed session lengths. This will be illustrated more in Chapter 4. Figure 2.9 shows the histogram for the sessions’ lengths before filtering. It is clear that most of the sessions are of length 15 or less and sessions with larger lengths are considered outliers. So, we grouped sessions into 15 different sessions’ groups, where each session group has sessions with the same length.
2.5 Modeling website parameters

In this section, we discuss the statistical analysis methods applied to the experimental results described earlier in Section 2.4. In modeling, we emphasize on the website parameters on which our active user-based user identification algorithm described in Section 2.3.3 depends.

2.5.1 Distribution functions

We selected three distribution functions as candidates to adequately represent the three website parameters on which our active user-based user identification depends. These are *number of records per user*, *inactive user time*, and *recorded records per second* in the web log. These three distribution functions are.

1. The power fit, which is given by

   \[ F(x) = ax^b \]  \hspace{1cm} 2.1

2. The reciprocal quadratic, which is given by

   \[ F(x) = \frac{1}{a + bx + cx^2} \]  \hspace{1cm} 2.2
3. The geometric fit, which given by

\[ F(x) = ax^b \]  \hspace{1cm} 2.3

2.5.2 Analytical results

One measure of the "goodness of fit" is the correlation coefficient. To explain the meaning of this measure, we consider the standard error, which quantifies the spread of the data around the mean, as follows

\[ S_t = \sum_{i=1}^{n_{\text{point}}} (\bar{y} - y_i) \]  \hspace{1cm} 2.4

where the average of the data points (\( \bar{y} \)) is simply given by

\[ \bar{y} = \frac{1}{n_{\text{point}}} \sum_{i=1}^{n_{\text{point}}} y_i \]  \hspace{1cm} 2.5

The quantity \( S_t \) considers the spread around a constant line (the mean) as opposed to the spread around the regression model. This is the uncertainty of the dependent variable prior to regression. We also consider the deviation from the fitting curve as

\[ S_r = \sum_{i=1}^{n_{\text{point}}} (y_i - f(x_i))^2 \]  \hspace{1cm} 2.6

Note the similarity of equation 2.6 to the standard error of the estimate given in equation 2.4. This quantity measures the spread of the points around the fitting function. Thus, the improvement (or error reduction) due to describing the data in terms of a regression model can be quantified by subtracting the two quantities presented in equations 2.4 and 2.6. Because the magnitude of the difference is dependent on the scale of the data, this difference is normalized to yield

\[ r \equiv \sqrt{\frac{S_t - S_r}{S_t}} \]  \hspace{1cm} 2.7
where $r$ is the correlation coefficient. As the regression model better describes the data, the correlation coefficient will approach unity. For a perfect fit, the standard error of the estimate will approach $S_r = 0$ and the correlation coefficient will approach $r=1$.

Next we model three website parameters *number of records per user*, *inactive user time* and *recorded records per second* in the web log. These three parameters determine the speed of the active user based algorithm presented in Section 2.3.2.3.

2.5.2.1 Modeling number of records per user

Figure 2.10 shows the probability of the number of records per user fitting. Table 2.4 shows the correlation coefficient for different models for the number of records per user probability. The power fit model is the most appropriate to fit the data since it has the closest value to one.
Table 2.4 Correlation coefficient for different models for the number records per user probability

<table>
<thead>
<tr>
<th>Model</th>
<th>Correlation coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power fit</td>
<td>0.41</td>
</tr>
<tr>
<td>Reciprocal quadratic</td>
<td>≈0.0</td>
</tr>
<tr>
<td>Geometric fit</td>
<td>≈0.0</td>
</tr>
</tbody>
</table>

2.5.2.2 Modeling inactive user time

Figure 2.11 shows the probability of inactive user time in seconds. Table 2.5 shows the correlation coefficient for different models for inactive user time probability. The reciprocal quadratic model is the most appropriate model to fit the data since its correlation coefficient is the closest to the value of one.

Table 2.5 Correlation coefficient for different models for inactive user time probability

<table>
<thead>
<tr>
<th>Model</th>
<th>Correlation coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power fit</td>
<td>≈0.0</td>
</tr>
<tr>
<td>Reciprocal quadratic</td>
<td>0.92</td>
</tr>
<tr>
<td>Geometric fit</td>
<td>0.11</td>
</tr>
</tbody>
</table>
2.5.2.3 Modeling recorded records per second

Figure 2.12 shows the probability of recorded records per second. Table 2.6 Correlation coefficient for different models for recorded records per second probability. The geometric fit model is the most appropriate to fit the data since its correlation coefficient has the value of one.

![Figure 2.12 Probability of recorded records per second](image)

Table 2.6 Correlation coefficient for different models for recorded records per second probability

<table>
<thead>
<tr>
<th>Model</th>
<th>Correlation coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power fit</td>
<td>0.89</td>
</tr>
<tr>
<td>Reciprocal quadratic</td>
<td>0.42</td>
</tr>
<tr>
<td>Geometric fit</td>
<td>1.00</td>
</tr>
</tbody>
</table>

2.6 Summary

In this chapter, we presented new techniques for preprocessing web log data including identifying unique users and sessions. We presented a fast active user-based user identification algorithm with time complexity of $O(n)$. For session identification we
presented an ontology-based session identification algorithm that uses the website structure to identify users’ sessions. We showed that the user identification algorithm depends on three website parameters: number of records per user, inactive user time and number of recorded records per second in the web log. In the chapters to follow, the output of this preprocessing step will be used as an input for different data mining techniques. In our research we focus on the clustering algorithm along with different learning algorithms for clustering results presentation.
CHAPTER III
MULTIDIMENSIONAL SESSIONS COMPARISON METHOD USING DYNAMIC PROGRAMMING

In this chapter, we present a new Multidimensional Sessions Comparison Method (MSCM) using dynamic programming. Our method takes into consideration different session dimensions, such as the page list, the time spent on each page and the length of each session. This is in contrast to other algorithms that treat sessions as sets of visited pages within a time period and don’t consider the sequence of the click-stream visitation or the session length.

3.1 Introduction

The problem of sequence comparison, which is defined as the measure of how much two or more sequences are similar to each other, has attracted researchers in different fields such as molecular biology [23], speech recognition [24], string matching [25] and traffic analysis studies [26].

In molecular biology, macromolecules are considered as long sequences of subunits linked together sequentially. Comparing these sequences helps to answer important questions in biology.

In speech recognition studies, speech is converted to a vector function of time, which is considered a continuous sequence. Sequence comparison can be used in
different applications such as recognizing an isolated work selected from limited vocabulary.

String matching represents each string as a sequence of characters. Sequence comparison can be used in spell checking in word processing applications.

In the context of web usage mining, measuring similarities between web sequences, or simply sessions, is an important step in the clustering process since the clustering process is grouping together similar web sessions.

In this chapter, we introduce a new method for measuring dissimilarities between web sessions that takes into account the sequence of events in a click stream visitation, the time spent in each event, and the length of the sessions. This method is used in our clustering method discussed in the next chapter.

In the next section, we provide some necessary definitions. In Section 3, we present the problem statement. In Section 4, we present related work on sequence comparison. In Section 5, we present previous work done on session comparison. In Section 6 we present our new Multidimensional Session Comparison Method (MSCM). In Section 7, we present experimental results and analysis. In Section 8, we present summary conclusions.

3.2 Definitions

We define the list of sessions $S = \langle s_1, \ldots, s_k \rangle$ where $s_i = \langle \langle p_1, \ldots, p_m \rangle, \langle t_1, \ldots, t_m \rangle \rangle$ is defined by two lists of entities: the first list consists of $m$ pages $\langle p_1, \ldots, p_m \rangle$ and the second list consists of $m$ time values $\langle t_1, \ldots, t_m \rangle$. The list of pages represents the pages visited by the user and the list of time values represents the time spent at each page. We also define the operator $|s|$ that returns the number of items in the sequence $s$. We also
define two functions \( s.p(x) \) and \( s.t(x) \) that return the page and the time spent at position \( x \), respectively.

3.3 Problem statement

The objective is to find a distance function, \( D \), defined over \( S \times S \), where \( D(s_i, s_j) \) is a numeric value that shows the extent to which sessions \( s_i \) and \( s_j \) are similar.

3.4 Related work

In this section, we present the well-established algorithms that can be used in the context of session comparison. Most of these algorithms were presented in the field of string matching.

3.4.1 Exact sequence matching

In this method, the distance function is defined as a Boolean function where the function returns \( \text{true} \) when there is an exact match between \( s_i \) and \( s_j \) and returns \( \text{false} \) otherwise, or expressed as follows:

\[
D(s_i, s_j) = \begin{cases} 
\text{true} & \text{if } s_i.p(x) = s_j.p(x) \text{ and } s_i.t(x) = s_j.t(x) \text{ and } |s_i| = |s_j|, \forall x \leq \max \{|s_i|, |s_j|\} \\
\text{false} & \text{otherwise}
\end{cases} 
\]

3.4.2 Approximate one dimension sequence matching

The idea behind approximate one dimensional sequence matching is based on limiting the sequence definition to one list of entities. For the web sessions case, the
sequence definition is limited to the list of pages. The matching is defined by a numeric value that is greater than or equal to zero. A value of zero represents an exact match, and the value increases as the difference between the sequences increases.

There are two major ways to compare the sequences. One is based on measuring the differences of the sequences’ items, while the other is based on measuring the similarities of the sequences’ items.

3.4.2.1 Measuring difference distance

The distance between sessions $s_i$ and $s_j$ is defined by the number of edit operations needed to transform $s_i$ to $s_j$. These operations are insertion (I), deletion (D), replacement (R), or no operation (M). For example, if we have two sequences $s_1$ and $s_2$ given as

$$s_1 = \langle p_0, p_2, p_5, p_3 \rangle$$

$$s_2 = \langle p_0, p_1, p_2, p_5, p_4 \rangle .$$

To transform $s_1$ to $s_2$, the following operations need to be applied:

M: no operation since $p_0$ ’s are matched in the first position of $s_1$ and $s_2$.

I: insertion of $p_1$ into the second position of $s_1$.

M M: $p_2$’s, $p_5$’s match on the third and fourth positions.

R: replacement of $p_3$ at last position of $s_1$ by $p_4$.

The dynamic programming method [27] can be used to find the minimum number of operations. For $s_i$ and $s_k$, $D(i, j)$ is defined to be the edit distance (number of edit operations) to convert the first $i_m$ characters of $s_i$ to the first $j_m$ characters of $s_k$.

The recursion base conditions are
\[ D(i,0) = i \] 3.2

and

\[ D(0, j) = j \] 3.3

The recurrence relation for \( D(i, j) \) is

\[ D(i, j) = \min[D(i-1, j)+1, D(i, j-1)+1, D(i-1, j-1)+t(i, j)] \] 3.4

where \( t(i, j) \) is defined to have value of 1 if and only if the \( i_{th} \) character of \( s_i \) and the \( j_{th} \) character of \( s_k \) are different, otherwise it has a value of 0.

In this approach, \( D(i, j) \) is first computed for the smallest possible values for \( i \) and \( j \). Typically, this computation is organized with a dynamic programming table of size \((n+1) \times (m+1)\). The table holds the values of \( D(i, j) \) for all the choices of \( i \) and \( j \). In the table, the vertical axis represents \( s_i \) and the horizontal axis represents \( s_k \). Because \( i \) and \( j \) begin at zero, the table has a zeroth row and a zeroth column. The values in the zeroth row and the zeroth column are filled indirectly from the base conditions for \( D(i, j) \). After that, the remaining \( n \times m \) subtable is filled in one row at time, in order of increasing \( i \). Within each row, the cells are filled in order of increasing \( j \). Except for the base case, the other \( D(i, j) \)'s are known once \( D(i-1, j-1), D(i, j-1), \) and \( D(i-1, j) \) have been computed. So the entire table can be computed in one row at a time, in the order of increasing \( i \), and in each row the values can be computed in order of increasing \( j \).
3.4.2.2 Measuring similarity distance

In a pairwise scores matrix, \( s(x, y) \) denotes the score obtained by aligning pages in \( s_i \) and \( s_j \). The alignment is done by inserting spaces (or no action) between the pages. A pairwise scores matrix sets the score \( s(x, y) \) to be greater than or equal to zero when the pages are the same, and less than zero if they mismatch.

The alignment value \( A \) is defined as

\[
A = \sum_{i=1}^{l} s(S'_i(i), S'_j(i))
\]

where \( S'_i(i) \) and \( S'_j(i) \) denote \( s_i \) and \( s_j \) after alignment.

For example, if we have \( P = \langle p_0, p_1, p_2, p_3, p_4 \rangle \), we can define the pairwise scores between different pages as shown in Table 3.1. If we have two sequences \( s_i \) and \( s_j \) as described in Table 3.2, the alignment value, \( A \), is calculated using equation 3.5, which gives the following result:

\[
A = -2 + 2 -1 + 0 - 3 - 2 - 4 + 2 = -8
\]

Table 3.1 Pairwise scores between different pages

<table>
<thead>
<tr>
<th></th>
<th>( p_0 )</th>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>( p_3 )</th>
<th>( p_4 )</th>
<th>( - )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_0 )</td>
<td>1</td>
<td>-2</td>
<td>-4</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
</tr>
<tr>
<td>( p_1 )</td>
<td>2</td>
<td>-3</td>
<td>-1</td>
<td>-3</td>
<td>-2</td>
<td></td>
</tr>
<tr>
<td>( p_2 )</td>
<td>2</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p_3 )</td>
<td>0</td>
<td>-4</td>
<td>-4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p_4 )</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( - )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
3.5 Previous work

In this section, we present an overview of the previous work on session comparison done for web usage mining. Most of the similarity measures used to compare sessions in web usage mining were simply based on intersections between the sets, such as the cosine measure or the Jaccard coefficient [8]. For example, Foss et al. [8] applied the Jaccard coefficient, which basically measures the degree of common visited pages in the compared sessions. This method does not take into consideration the sequence of events. So, the algorithm does not differentiate between the following two situations, \( p_0 \) is visited before page \( p_1 \), or \( p_1 \) is visited before \( p_0 \). Path Feature Space [28] were used to represent all the navigation paths. The similarity between each two paths was measured by the definition of path angle. In the path angle method, each navigation path is represented as a vector and the similarity between paths is the cosine similarity between the vectors.

A non-Euclidean distance measure was presented using the sequence alignment method (SAM) [29-31], which is derived from the Levenshtein [32] approach, and it takes into account the weight of different operations. The distance formula is defined by

\[
D(s_i, s_j) = \min(D * w_d + I * w_i + R * w_r)
\]

where \(D, I, R\) are the number of deletion, insertion and replacement operations, respectively, needed to convert \(s_i\) to \(s_j\); and \(w_d, w_i, w_r\) are the weights of these operations, respectively. Unlike the Levenshtein method, SAM is done in two steps.

### Table 3.2 Two sequences \(s_i\) and \(s_j\)

<table>
<thead>
<tr>
<th>(s_i)</th>
<th>(p_0)</th>
<th>(p_2)</th>
<th>(-)</th>
<th>(p_3)</th>
<th>(p_2)</th>
<th>(p_0)</th>
<th>(p_3)</th>
<th>(p_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s_j)</td>
<td>(p_1)</td>
<td>(p_2)</td>
<td>(p_0)</td>
<td>(p_3)</td>
<td>(-)</td>
<td>(p_4)</td>
<td>(-)</td>
<td>(p_1)</td>
</tr>
</tbody>
</table>
First, it reorders the common elements such that the common elements in the two sequences appear in the same order. In the second step, it inserts the uncommon elements in both sequences so they appear the same.

The multidimensional sequence alignment method MDSAM [29] is a modified version of the sequence alignment method where it ultimately finds the set of operations inducing the possibly smallest sum of multidimensional operational costs. The full algorithm description can be found in [29].

3.5.1 Limitations of the previous work

The algorithms that use the Euclidean distance for vector or the cosine measure have several limitations [33]:

1. The transferred space could be of very high dimension.
2. The original click stream is naturally a click sequence which cannot be fully represented by a vector or a set of URLs where the order of clicks is not considered;
3. Euclidean distance has been proven in practice to be not suitable for measuring similarity in categorical vector space. The multidimensional algorithms do not solve the \textit{inter-attribute relationship} problem, which is defined as the problem of considering the relationship between the attributes in different dimensions.

3.6 Multidimensional session comparison method (MSCM)

In this section, we present our new Multidimensional Session Comparison Method (MSCM). Next, we present assumptions on which we based our algorithm. After constructing the algorithm, we present a detailed description of the algorithm. Finally, we present a time complexity analysis for the algorithm.
3.6.1 Assumptions

We assume that three edit operations are allowed to convert one session to another: deletion, insertion and swap. The deletion operation $D(x)$ is defined as deleting an event in the session at position $x$. The insertion operation $I(x)$ is defined as inserting a new event in the session at position $x$. The swap operation $S(x)$ is defined as swapping between events in the session at positions $x$ and $x+1$.

We present the following assumptions about the sessions and our algorithm:

1. For the two dimension—the pages list and time list—we assume that the page list is a primary dimension and the time list is a secondary list.

2. The navigation behavior for the user is determined mainly by the primary dimension, which is the page list.

3. The first dimension, page list, is a nominal attribute and the other dimension, time value list, is a continuous attribute. So, the difference between pages $p_1$ and $p_2$ is considered the same as the difference between pages $p_1$ and $p_{100}$, but the difference between $t = 1$ and $t = 7$ is not the same as the difference between $t = 1$ and $t = 100$.

4. The distance $d_{mscm}(s_i, s_j)$ between two sessions $s_i$ and $s_j$ is directly proportional to the minimum number of edit operations needed to convert $s_i$ to $s_j$.

5. The distance between two sequences is inversely proportional to the maximum length of the compared sequences.

6. The weight of the swap operation is directly proportional with the time spent on the first page.
3.6.2 Algorithm construction

Based on the first four assumptions, we present a one dimensional distance function defined by

\[ d_{mscm}(s_i, s_j) \propto \min(D \cdot w_d + I \cdot w_i + S \cdot w_s) \] 3.7

where:

- \( d_{mscm} \) is the edit distance based on MSCM
- D is the number of deletion operations
- I is the number of insertion operations
- S is the number of swap operations
- \( w_d \) is the weight of the deletion operation
- \( w_i \) is the number of insertion operation
- \( w_s \) is the weight of swap operation

Based on the fifth assumption, the total distance divided by the maximum length of both sequences is

\[ d_{mscm}(s_i, s_j) \propto \frac{\min(D \cdot w_d + I \cdot w_i + S \cdot w_s)}{\max(|s_i|, |s_j|)} \] 3.8

where

- \(|s|\) is the length of the sequence.

Based on the final assumption, the weight of the swap operation multiplied by the Heaviside step function \( \Phi(t) \), which is defined as

\[ \Phi(t) = \begin{cases} 0 & t \leq 0 \\ 1 & t > 0 \end{cases} \] 3.9

where \( t \) is the time spent on the page on which swap operation is performed, \( s \) given as
\[ d_{mscm}(s_i, s_j) = \frac{\min(D \cdot w_d + I \cdot w_i + S \cdot w_s \cdot \Phi(t))}{\max(|s_i|, |s_j|)} \] 3.10

The distance function defined in equation 3.10 takes into consideration the two dimensions in the web session and gives a proper solution to the inter-attribute relationship, unlike other algorithms where the inter-attribute relationship is solved by computing trajectory between the first and the second attributes. Also, the distance calculated in equation 3.10 is considered to be the absolute distance since it is relative to the maximum length of the sequences.

3.6.3 Algorithm description

The algorithm used for finding the minimal edit operation is based on the dynamic programming in [34, 35]. Table 3.3 summarizes the MSCM algorithm’s major steps to find the minimum number of edit operations.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1 | Set n to be the length of \(s_1\).  
Set m to be the length of \(s_2\).  
If \(n = 0\), return m and exit.  
If \(m = 0\), return n and exit.  
Construct a matrix containing 0..m rows and 0..n columns. |
| 2 | Initialize the first row to 0..n.  
Initialize the first column to 0..m. |
| 3 | Examine each character of \(s_1\) (i from 1 to n). |
| 4 | Examine each character of \(s_2\) (j from 1 to m). |
| 5 | If \(s_1[i]\) equals \(s_2[j]\), the cost is 0.  
If \(s_1[i]\) doesn't equal \(s_2[j]\), the cost is 1. |
| 6 | Set cell \(d[i,j]\) of the matrix equal to the minimum of:  
a. The cell immediately above plus 1: \(d[i-1,j]\) + 1.  
b. The cell immediately to the left plus 1: \(d[i,j-1]\) + 1.  
c. The cell diagonally above and to the left plus the cost: \(d[i-1,j-1] + \text{cost}\). |
| 7 | After the iteration steps (3, 4, 5, 6) are complete, the distance is found in cell \(d[n,m]\). |
For example, if we have two web sessions:

\[ s_0 = \{\langle p_{14}, p_{11}, p_6, p_{10}, p_1, p_{11}, p_8 \rangle, \langle 1, 2, 4, 3, 2, 4, 5 \rangle\}, \text{ and} \]

\[ s_1 = \{\langle p_{15}, p_6, p_7, p_{10}, p_7, p_1, p_{11} \rangle, \langle 2, 4, 6, 3, 1, 3, 5 \rangle\} \]

Table 3.4 shows the matrix to be used to compute the minimum edit distance between sequences \( s_0.p \) and \( s_1.p \).

Table 3.4 Matrix used to compute the minimum edit distance, when only the zeroth column and row are filled in

<table>
<thead>
<tr>
<th>( d_{max} )</th>
<th>( p_{14} )</th>
<th>( p_{11} )</th>
<th>( p_6 )</th>
<th>( p_{10} )</th>
<th>( p_1 )</th>
<th>( p_{11} )</th>
<th>( p_8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i )</td>
<td>( j )</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>( p_{15} )</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p_6 )</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p_7 )</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p_{10} )</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p_7 )</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p_1 )</td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p_{11} )</td>
<td>7</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The \((i,j)\) cell in Table 3.4 stands for the minimum edit operations to convert the first \( i \) pages in sequence \( s_0 \) to the first \( j \) pages in sequence \( s_1 \). Taking into consideration the zero length sequence, the table size will be the sequence length plus 1. So for the example in Table 3.4, we have table of size \((7+1) \times (7+1)\). In the table, the vertical axis represents \( s_0 \) and the horizontal axis represents \( s_1 \). The values in row zero and column zero are filled in directly from the base conditions in equations 3.2 and 3.3, respectively. After that, the remaining cells in the table are filled in one row at time, in order of increasing \( i \) and then in order of increasing \( j \), using the recurrence equation 3.4. Within
each row, the cells are filled in order of increasing \( j \). Table 3.5 shows when the entire cells are filled in. The minimum edit distance is the last cell value, which in our example equals to 5.

Table 3.5 Matrix used to compute the minimum edit distance, when the entire cells are filled in

<table>
<thead>
<tr>
<th>( d_{m_{scm}} )</th>
<th>( P_{14} )</th>
<th>( P_{11} )</th>
<th>( P_{6} )</th>
<th>( P_{10} )</th>
<th>( P_{1} )</th>
<th>( P_{11} )</th>
<th>( P_{8} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

3.6.4 Time complexity analysis

The time complexity analysis for the dynamic programming table for computing the minimum edit distance is reported to be \( O(m \cdot n) \)[34], where \( m \) is the length of the first session and \( n \) is the length of the second session. This is straightforward since it needs \( (n \cdot m) \) steps to fill the table for sessions of length \( n \) and \( m \), respectively.

3.7 Experimental results and analysis

In this section we present a few experiments that show how MSCM provides better results than other sessions comparison methods. We compare the results with other three methods: Euclidean based distance (Path Feature Space [28]) method, sequence alignment method SAM [30], and Multidimensional Sequence Alignment Method
MDSAM [29]. More experimental results are presented in the next chapter, which is the output of the clustering algorithm that adopts MSCM as its distance function.

For our experimental results, assume we have the following sessions:

\[
s_0 = \{\langle p_{15}, p_7, p_6, p_{10}, p_1, p_7, p_{11}\rangle, \langle 3,0,2,4,0,4,5\rangle\}
\]

\[
s_1 = \{\langle p_{15}, p_6, p_7, p_{10}, p_7, p_1, p_{11}\rangle, \langle 3,4,1,4,1,1,6\rangle\}
\]

\[
s_2 = \{\langle p_{14}, p_{11}\rangle, \langle 4,5\rangle\}
\]

\[
s_3 = \{\langle p_5, p_7\rangle, \langle 5,4\rangle\}
\]

\[
s_4 = \{\langle p_{15}, p_7, p_6, p_{10}, p_1, p_7, p_{11}\rangle, \langle 3,2,1,4,3,4,5\rangle\}
\]

\[
s_5 = \{\langle p_{15}, p_7, p_6, p_{10}, p_1, p_3, p_1\rangle, \langle 2,3,2,4,3,7,5\rangle\}
\]

Table 3.6 summarizes the distance between different sessions using different session comparison methods.

<table>
<thead>
<tr>
<th>Sessions</th>
<th>Method</th>
<th>Path Feature Space</th>
<th>SAM</th>
<th>MDSAM</th>
<th>MSCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s_0, s_1)</td>
<td>(\sqrt{5^2 + 4^2} = 6.4)</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>(s_2, s_3)</td>
<td>(\sqrt{2^2 + 2^2} = 2)</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(s_4, s_5)</td>
<td>(\sqrt{2^2 + 3^2} = 3.6)</td>
<td>2</td>
<td>5</td>
<td>0.29</td>
<td></td>
</tr>
</tbody>
</table>

Sessions \(s_0\) and \(s_1\) are almost the same except for the swap between pages \(p_6, p_7\) and pages \(p_1, p_7\). However, the time spent on \(p_7\) and \(p_1\) in \(s_1\) is zero. This distance—i.e., time spent on a page is zero—arise because the time resolution for recording a web record is one second. Therefore, we cannot tell which page is loaded first, and so the order of these pages should be ignored. The other three algorithms do not recognize this, even the multidimensional ones, and all of them indicate that there is a difference between the sessions where there is actually no difference as indicated from MSCM.
As for sessions $s_2$ and $s_3$, it is obvious that they are not similar at all. The other three methods measure the difference based on the edit distance and they give results that do not reflect the complete mismatch between the sequences. On the contrary, MSCM computes the absolute difference, which is the edit distance divided by the maximum sessions’ length. Thus, for sessions $s_2$ and $s_3$, MSCM returns the value of one, which indeed reflects the complete mismatch.

The usefulness of measuring the absolute value can also be seen when we consider the results of comparing $s_2$ and $s_3$ versus comparing $s_4$ and $s_5$. The other three methods show almost the same degree of differences when comparing sessions $s_2$ and $s_3$ versus comparing sessions $s_4$ and $s_5$ but this is not true at all because sessions $s_4$ and $s_5$ are almost the same except for the last two pages, while sessions $s_2$ and $s_3$ are completely different. On the other hand, MSCM algorithm recognizes that the degree of difference between sessions $s_2$ and $s_3$ (returning a value of 1) is not the same as the degree of difference between sessions $s_4$ and $s_5$ (returning a value of 0.29).

3.8 Summary and conclusion

In this chapter we presented a new Multidimensional Session Comparison Algorithm (MSCM), which is based on dynamic programming. Unlike other methods, MSCM takes into consideration other dimensions in the session, such as the time spent on the page and the total session length. The methods showed more accurate results than other known methods in comparing web sessions, such as Sequence Alignment Method (SAM), Multidimensional Sequence Alignment Method (MDSAM), and Path Feature Space. The output of the MSCM is presented in the form of dissimilarity matrix, which
can be used by different clustering techniques, such as the hierarchal, the k-mean, and the equivalence classes clustering algorithms.
CHAPTER IV

ENHANCING WEBSITE STRUCTURE BY MEANS OF HIERARCHAL CLUSTERING ALGORITHMS AND ROUGH SET LEARNING APPROACH

4.1 Introduction

In this chapter, we present a new way to enhance the website structure by means of hierarchal clustering algorithms and rough sets learning approach. Figure 4.1 shows the system workflow. The workflow starts by clustering the web sessions into different clusters using the dissimilarity matrix. The clustering results are then presented in the form of examples as shown in Table 4.1, where each web session along with its clustering result represents one example in the examples table.

<table>
<thead>
<tr>
<th>Example No.</th>
<th>1st Page</th>
<th>2nd Page</th>
<th>3rd Page</th>
<th>Clustering Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>p₀</td>
<td>p₁</td>
<td>p₄</td>
<td>C₁</td>
</tr>
<tr>
<td>2</td>
<td>p₂</td>
<td>p₃</td>
<td>p₅</td>
<td>C₂</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>N</td>
<td>p₂</td>
<td>p₃</td>
<td>p₅</td>
<td>Cₖ</td>
</tr>
</tbody>
</table>

The examples are then divided into two independent sets. The first set is used by different classifiers to learn rules that describe the system; the rules are presented in the if then format. For example the following two rules can be learned from the first two examples in Table 4.1.

if 1st page = p₀ and 2nd page = p₁ and 3rd Page = p₄ then Cluster = C₁
\[
\text{if } 1^{st} \text{ page } = p_2 \text{ and } 2^{nd} \text{ page } = p_3 \text{ and } 3^{rd} \text{ Page } = p_5 \text{ then Cluster } = C_2
\]

The second set along with the learned rules from the first set are used in the inference engine to estimate the accuracy of the classification process. The clustering results along with the generated rules are then incorporated to enhance the structure of the website.

![Diagram of Web usage classification and prediction workflow](image)

Figure 4.1 Web usage classification and prediction workflow

The rest of the chapter is organized as follows. In Section 2, we present an overview of clustering analysis. In Section 3, we present two algorithms for clustering web sessions. In Section 4, we present two different classifiers to describe and predict a web session’s classes. In Section 5, we present a method to estimate the accuracy of different
classifiers. In Section 6, we present experimental results. In Section 7, we show how the results are incorporated in enhancing the website structure. In Section 8, we present results discussion. In Section 9, we present summary and conclusion.

4.2 Clustering analysis

Clustering is a useful technique for grouping objects such that objects within a single group have similar characteristics, while objects in different groups are dissimilar. In the context of web usage mining, the objects are users’ sessions. Each session contains the pages visited by the user at a certain time. Clusters can be used to cluster the users such that users with the same browsing behavior are in single cluster. For example, one cluster may consist of predominantly freshman students who register for classes, while another may consist of professors who upload their classes’ grades. The clusters can then be used to identify dominant browsing behaviors, evaluate the website structure and predict users’ browsing behavior. Clustering web usage sessions’ is an example of clustering where objects are of a non-numeric data type such as nominal or categorical data type.

4.2.1 Clustering algorithms

Clustering algorithms can be classified into \textit{partitional clustering} and \textit{hierarchal clustering} [36, 37]. Partitional clustering algorithms divide \( n \) objects into \( k \) clusters that satisfy two conditions (1) each cluster contains at least one object, and (2) each object exactly belongs to one cluster. Equation 4.1 shows one of the commonly used criterion functions

\[
E = \sum_{i=1}^{k} \sum_{x \in C_i} d(\bar{x}, \bar{m}_i) \tag{4.1}
\]
In equation 4.1, \( \bar{m}_i \) is the centroid of cluster \( C_i \), while \( d(\bar{x}, \bar{m}_i) \) is the Euclidean distance between \( \bar{x} \) and \( \bar{m}_i \) defined in equation 4.2

\[
d(\bar{x}, \bar{m}_i) = \left( \sum_{i=1}^{d} (\bar{x} - \bar{m}_i)^2 \right)^{1/2}
\]

4.2

The criterion function \( E \) attempts to minimize the distance of every object from the mean of the cluster to which the object belongs. One of the common approaches to minimize the criterion function is the iterative k-means method. While the use of the k-means method could yield satisfactory results for numeric attributes, it is not appropriate for data sets with categorical attributes [38], as it is the case in web sessions.

Hierarchical clustering algorithms work by grouping data objects into a tree of clusters. A hierarchical method can be classified as agglomerative or divisive. Agglomerative hierarchical clustering, which is the most common strategy, starts by placing each object in one cluster then merges similar objects together until forming one cluster that has all the objects in it or some other termination condition exists. Divisive hierarchical clustering starts with all objects in one cluster and divide them up until each object forms a cluster by itself or some other termination condition is met.

At the first step of the agglomerative method, the dissimilarity matrix can be used to determine how close the objects are to one another. Once the first step is completed and the first level of clusters is generated, there will be a need to compare the clusters rather than comparing the objects. Next, we present the five most common techniques to measure the difference between clusters:

- **Single linkage**: the distance between any two clusters is the shortest distance from any object in one cluster to any object in the other [39].
• **Complete linkage**: the distance between any two clusters is the farthest distance from any object in one cluster to any object in the other.

• **Average linkage**: the distance between any two clusters is the average distance from all objects in one cluster to all individuals in the other.

• **Ward’s method**: the distance between two clusters is the sum of squares of the distance between all objects in both clusters [40].

• **Centroid method**: the distance between two clusters is the distance between their centroids.

4.2.2 Properties of agglomerative hierarchal clustering techniques

The single linkage method tends to have the chaining property [41]. As shown in Figure 4.2, chaining is producing two well separated clusters with an intermediate chain of data.

![Figure 4.2 Two well separated clusters with intermediate chain](image)

Previous empirical investigations indicate that the average linkage method and the Ward’s method had superior performance. For example, Cunningham and Ogilvie [42] compared several hierarchal techniques and found that the average linkage method
performs most satisfactory for the data sets they considered. Kuiper and Fisher [43] investigated six hierarchal techniques and found that the Ward’s method classifies the data very well. Finally, Blashfiled [44] compared the single linkage method, the complete linkage method, the average linkage method, and the Ward’s method using a quantifying statistical method explained in [45]. They found that the Ward’s method performed very well over the other methods. Later in Section 4.3.6, we explain why we use the Ward’s method over the average linkage method.

4.3 Clustering web sessions

Clustering web sessions is to group similar website usage behavior together. The clustering results can be used to identify dominant browsing behavior, evaluate the website structure and predict users’ browsing behavior. We present two clustering algorithms: hierarchal and equivalence classes clustering. Both algorithms have the ability to deal with nominal attributes, such as session with different sizes, and both have the ability to adopt different dissimilarity matrixes. Unlike the hierarchal clustering algorithm, clusters generated from the equivalence classes clustering algorithm do not depend on the seed where the clustering process starts. In this section, we next present several definitions along with a formal description of the problem statement. At the end of the section, we present the two proposed algorithms along with their running time complexity analysis.

4.3.1 Definitions

Web sessions $S$ is defined as $S = \langle s_1, \ldots, s_k \rangle$, where $k$ is the number of sessions. Each web session $s_i$ is defined as $s_i = \langle p_{i,1}, \ldots, p_{i,n} \rangle$, where $n$ is the number of pages in session $s_i$, and $p_{i,k}$ is the $k_{th}$ page in session $s_i$. Web sessions clusters $C$ is defined
as $C = \langle c_1, \ldots, c_m \rangle$, where $m$ is the number of clusters. Each cluster $c_j$ is defined as $c_j = \langle s_{j,1}, \ldots, s_{j,l} \rangle$, where $l$ is the number of sessions in cluster $c_j$.

The dissimilarity function $\delta_{i,j}$ is defined over $S \times S$, where $0 \leq \delta_{i,j} \leq 1$; a value of one implies perfect dissimilarity, and a value of zero implies perfect similarity. A detailed description of this function was presented in Chapter 3. The cluster centroid $ce_i$ is defined as the session in the middle of the cluster $c_i$. The centroid defined in equation 4.3 uses the Ward’s method in defining the minimum distance. In Section 4.3.6, we explain why we use the Ward’s method over other methods, such as the average linkage method, in defining the minimum distance for the centroid.

$$ce_i = s_{i,k} \text{ where } s_{i,k} = \min \left( \sum_{j=1}^{n} \left( \delta_{j,k} \right)^2 \right), \forall s_{i,k} \in c_i \tag{4.3}$$

We also define an overloaded version of the dissimilarity function defined earlier in equation 3.10 to be applied to the clusters. The overloaded version of $\delta_{i,j}$ shown in equation 4.4 accepts clusters as inputs and uses the centroid-method in determining the difference between clusters.

$$\delta_{i,j} = \sum_{\text{cluster } i} \sum_{\text{cluster } j} \delta_{ce_i,ce_j} \tag{4.4}$$

Finally, we define the threshold $\lambda$ as the maximum difference allowed between sessions in the same cluster.
4.3.2 Problem statement

Given web sessions $S$, dissimilarity function $\delta_{i,j}$, and threshold value $\lambda$, the objective is to find web session’s clusters $C_{\lambda}$’s such that for $\forall c_k, c_k \in C_{\lambda}$ it is true that $\delta_{i,j} \leq \lambda$ for $\forall s_i, s_j \in c_k$.

4.3.3 Hierarchal clustering algorithm

Figure 4.3, shows the clustering algorithm used to cluster the web sessions. Initially, each session is placed in a cluster by itself and the threshold value $\lambda$ is initialized to zero. Then, the dissimilarity value between all pairs of clusters is checked. If the dissimilarity value is less than or equal the threshold value, the two clusters are merged to form a new cluster. The new number of clusters is then checked. If it is found to be less than or equal to the minimum number of clusters the algorithm exits of the loop and the set of clusters is then returned (in Section 4.3.5 we discusses how to choose the minimum number of clusters). Otherwise, the threshold value $\lambda$ is incremented and the process is repeated again.
The time complexity analysis of the algorithm in Figure 4.3 shows that the algorithm has two loops, an inner and an outer loop. The outer loop has the worst case of \( n \) cycles in the event the initial number of the clusters is \( n \), while the inner loop has the worst case of \((n-1)\) cycles. In our work, we assumed that the dissimilarity function is provided in the form of a matrix, where element \( ij \) contains the dissimilarity between sessions \( i \) and \( j \). Therefore, the time complexity for finding the dissimilarity value is fixed with a constant time \( c \) that is independent of the initial number of clusters \( n \). Hence the overall complexity of the algorithm is \( O(n(n-1)c) = O(cn^2) = O(cn^2) \).

4.3.4 Equivalence classes clustering algorithm

We define an equivalence relation called \( \text{belongs to} \), \( \sim \), on \( C \times C \) that satisfies reflexive, symmetric and transitive properties. The reflexive property implies that \( c_i \sim c_i \) for \( \forall c_i \in C \). The symmetric property implies that if \( c_i \sim c_j \) then \( c_j \sim c_i \) for \( \forall c_{i,j} \in C \). The transitive property implies that if
$c_i \sim c_j$ and $c_j \sim c_k$ then $c_j \sim c_k$ for $\forall c_{i,j,k} \in C$. The first two properties (reflexive and symmetric) are satisfied in the hierarchal algorithm defined in Section 4.3.3. So, to achieve the equivalence relation defined earlier, we modify the algorithm described in Figure 4.3 to satisfy the equivalence relation.

Figure 4.4 shows the equivalence classes clustering algorithm that satisfies the three properties for the equivalence relation mentioned earlier. The algorithm is a modified version of the one showed in Figure 4.3 in which the merging between clusters is not done unless all pairs of sessions in both clusters has a dissimilarity value less than or equal to the threshold value $\lambda$.

The time complexity analysis for this algorithm is the same as the one for the hierarchal clustering algorithm shown in Figure 4.3, except for an extra third inner loop, which increases the complexity by one degree to become $O(n^3)$. 
Figure 4.4 Equivalence classes clustering algorithm

4.3.5 Determining a common termination condition for different sessions lengths

Prior to applying the web sessions to the clustering algorithms described earlier, the sessions are grouped in a way sessions with the same session length are grouped together. In the clustering algorithms shown in Figure 4.3 and Figure 4.4, the termination condition depends on the minimum number of clusters. To choose a minimum number of clusters that is common for all session’s length groups, the number of clusters is normalized to the number of clusters in the first iteration for each session length group. To illustrate this idea, consider for example Table 4.2 that shows how a group of web sessions divided
according to length of the session into two groups. The first group shows the number of clusters for session length of 3 at different clustering iterations. The second group shows the number of clusters for session length of 4 at different clustering iterations.

Table 4.2 Number of clusters for different session lengths at different iterations

<table>
<thead>
<tr>
<th>Session length</th>
<th>Iteration</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td>255</td>
<td>39</td>
<td>35</td>
<td>9</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>479</td>
<td>141</td>
<td>115</td>
<td>64</td>
<td>35</td>
<td>26</td>
<td>24</td>
<td>15</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

It is clear that the initial number of clusters is different in the two set of the group sessions. So, in order to choose the same termination condition we calculate the percentage of the number of the clusters from the initial number of clusters for different session lengths at different iterations. Table 4.3 shows the percentage of the number of the clusters from the initial number of clusters for different session lengths at different iterations. So, for example, if we choose to stop the clustering process when the percentage of the number of clusters is 14% from the initial number of clusters, then we stop at the iteration where the value of the percentage is the closest to 14%, which is, in this case, iterations 3 and 4 for session lengths 3 and 4, respectively.

Table 4.3 Percentage of the number of the clusters from the initial number of clusters for different session lengths at different iterations

<table>
<thead>
<tr>
<th>Session length</th>
<th>Iteration</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td>100%</td>
<td>15%</td>
<td>14%</td>
<td>3%</td>
<td>2%</td>
<td>2%</td>
<td>2%</td>
<td>2%</td>
<td>1%</td>
<td>0%</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>100%</td>
<td>29%</td>
<td>24%</td>
<td>13%</td>
<td>7%</td>
<td>5%</td>
<td>5%</td>
<td>3%</td>
<td>1%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

4.3.6 Ward’s method improves determining a common termination condition

The reason we use Ward’s method in defining the centroid in equation 4.3 over other methods, like average linkage, is because it shows a slower convergence, which
helps in determining a common termination condition for all session length groups more accurately. To illustrate why we want a slow convergence, first consider Figures 4.5 and 4.6, which show the percentage of the number of the clusters from the initial number of clusters for a specific session length at different iterations using the average linkage method and the Ward’s method, respectively. Next, assume we want to stop the clustering process when the number of clusters is 20% from the initial number of clusters (in Section 4.6 we explain how to choose these percentage points). Using the average linkage method, Figure 4.5 shows that the 20% point occurs between iterations 6 and 7, where the percentage is 23.50% and 14.53%, respectively. So, the closest iteration is iteration 6 giving a percentage of 23.50% and an error of 3.5%. When using the Ward’s method, as shown in Figure 4.6, the 20% point occurs between iterations 53 and 54, where the percentage is 20.5% and 19.7%, respectively. So, the closest iteration is 54 giving a percentage of 19.7% and an error of 0.3%. The error value represents how accurate the clustering process is with a common termination condition for all sessions with the different session lengths.
4.4 Web sessions’ classifiers

We present two methods for cluster classification. The first method is based on the centroid and the second is based on the inductive learning program BLEM2. The rules learned from both classifiers are used in both predicting and describing web sessions clusters. Table 4.4 shows an example of a rule generated by a web sessions’ classifier.
To predict web sessions cluster, the rule in Table 4.4 can be presented in an *if then* statement such as

\[
\text{If 1st Page}= p_{45} \text{ and 2nd Page}= p_{84} \text{ and 3rd page}= p_{204} \text{ then Cluster}= C_1
\]

So, for any new session in the web log the above rule can be applied in order to predict the session’s cluster. We use the classifier accuracy estimator described in Section 4.5 to estimate the accuracy of the prediction. Experimental results based on this method are presented in Section 4.6.

To describe a cluster, the physical page-name lookup-table is used to find the physical page name. For example, the physical name for pages in Table 4.4 is 

\[
\langle \text{signon, BeforeClassSearch, ClassSearch} \rangle
\]

So, the cluster can then be described as *Class Search* cluster, meaning it contains a group of users who are searching for classes. The evaluation of the cluster description is based on the length of the description. For this example, the length of the description is 3. In Section 4.6.4, we present experimental results for the description length using different classifiers.

4.4.1 The centroid approach

The centroid approach is based on using the cluster centroid to describe clusters and to predict users’ classes based on past behavior. As defined in equation 4.3 the centroid is the session that has the minimum average square distance between all other sessions in the same cluster.
To illustrate this, assume that cluster $c_k$ has session $s_i$ as its centroid, which represents the following page sequence:

$$s_i = \langle p_{45}, p_{84}, p_{204} \rangle .$$

From the page-name lookup-table assume we found that the sequence has the following physical page names:

$$s_i = \langle \text{signon}, \text{BeforeClassSearch}, \text{ClassSearch} \rangle .$$

The cluster can then be described as the Class Search cluster, meaning it contains a group of users who are searching for classes.

Beside using the centroid for describing the clusters, this description is also used to predict users’ classes. From the above example, the following rule can be generated

$$\text{if } s_{i,1} = p_{45} \text{ and } s_{i,2} = p_{84} \text{ and } s_{i,3} = p_{204} \text{ then } s_i \in c_k$$

The generated rules, like the one above, may be applied to an inference engine to predict the class of the future coming sessions.

4.4.2 Rough set approach

In this subsection, we present the use of the rough set learning program BLEM2 in classifying different users’ sessions. BLEM2 is an implementation of one of the LERS [46, 47] family learning programs which was introduced by Grzymala-Busse [48]. We use the information system notion presented by Pawlak [49,50] in which information system $S$ is defined as a pair $S = (U, A)$, where $U$ is a nonempty finite set of attributes in $A$, i.e., a vector of attribute values that denotes each object. Each attribute in $A$ is associated with a set of values called the domain of attribute.
Both hierarchal clustering algorithms described earlier produce a special case of an information system called the decision table. In a decision table, there is a designated attribute called the decision or class attribute, and other attributes called condition attributes.

Table 4.5 shows an example of a decision table produced by the clustering algorithm where the universe $U$ consists of 16 examples. In Table 4.5, the attribute $Cluster No.$ is the decision attribute and the attributes $Page1$, $Page2$ and $Page3$ are the condition attributes. The set of the decision attribute is \{0, 1, 2, 3, 4\} and the set of the condition attribute $A$ is \{45, 58, 84, 108, 120, 160, 186, 194, 204, 241, 251, 444, 463\}.

<table>
<thead>
<tr>
<th>Session No.</th>
<th>Page 1</th>
<th>Page 2</th>
<th>Page 3</th>
<th>Cluster No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45</td>
<td>444</td>
<td>108</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>444</td>
<td>108</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>45</td>
<td>444</td>
<td>84</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>45</td>
<td>444</td>
<td>463</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>160</td>
<td>241</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>45</td>
<td>160</td>
<td>108</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>45</td>
<td>160</td>
<td>241</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>45</td>
<td>80</td>
<td>194</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
<td>80</td>
<td>251</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>45</td>
<td>80</td>
<td>251</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>45</td>
<td>120</td>
<td>58</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>45</td>
<td>120</td>
<td>186</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>45</td>
<td>120</td>
<td>160</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>45</td>
<td>84</td>
<td>204</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>45</td>
<td>84</td>
<td>204</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>45</td>
<td>84</td>
<td>204</td>
<td>4</td>
</tr>
</tbody>
</table>

The partition on $U$ determined by the decision attribute $Cluster No.$ is

$C_0 = [1,2,3,4]$,  
$C_1 = [5,6,7]$,  
$C_2 = [8,9,10]$,  
$C_3 = [11,12,13,14,15,16]$.
$C_3 = [1, 12, 13]$, and 
$C_4 = [14, 15, 6]$

where $C_k$ is the set of sessions that belongs to cluster $k$. The rough set theory presents the concepts of lower and upper approximations in case of inconsistency (i.e., having more than one decision for the same condition value).

Let $A = (U, R)$ be an approximation space, where $U$ is set of objects and $R$ is an equivalence relation defined on $U$. Let $X$ be a nonempty subset of $U$. Then, the lower approximation of $X$ by $R$ in $A$ is defined as

$$\underline{RX} = \{e \in U \mid [e] \subseteq X\} \quad 4.5$$

and the upper approximation of $X$ by $R$ in $A$ is defined as

$$\overline{RX} = \{e \in U \mid [e] \cap X \neq \emptyset\} \quad 4.6$$

where $[e]$ denotes the equivalence classes containing $e$.

The boundary set of $X$ is defined as

$$BN_R(X) = \overline{RX} - \underline{RX} \quad 4.7$$

A subset $X$ of $U$ is said to be $R$-definable in $A$ if and only if $\overline{RX} = \underline{RX}$. The pair $(\overline{RX}, \underline{RX})$ defines a rough set in $A$, which is a family of subsets of $U$ with the same lower and upper approximations as $\overline{RX}$ and $\underline{RX}$.

From Table 4.5, the lower and upper approximation of $C_0, C_1, C_2, C_3$, and $C_4$ are:
As of the case in the previous example, the clustering algorithms presented earlier do not produce inconsistent rules and so the upper approximation is the same as the lower approximation; i.e., $\overline{RX} = RX$, and the boundary set is $BN_R(X) = RX - RX = \Phi$.

We use BLEM2 to learn rules from the lower approximation $AX_i$ since it is the same as the upper approximation $\overline{AX}_i$ and the boundary set $BN_A(X_i)$ is an empty set. The rules learned from the lower approximation are called certain rules. Table 4.6 shows the certain rules learned from Table 4.5 using BLEM2. In the rules table, the entry -1 denotes a “do not care” condition. The support column denotes the number of examples covered by the rule. The certainty column denotes the ratio of the examples that match both the rule and its decision value. The strength column is the support of the rule over the entire training set. The coverage column is the ratio of the decision value class covered by the rule.

<table>
<thead>
<tr>
<th>Page 1</th>
<th>Page 2</th>
<th>Page 3</th>
<th>Cluster</th>
<th>Support</th>
<th>Certainty</th>
<th>Strength</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>2</td>
<td>-1</td>
<td>1</td>
<td>288</td>
<td>1</td>
<td>0.0974</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>3</td>
<td>-1</td>
<td>2</td>
<td>118</td>
<td>1</td>
<td>0.0399</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>4</td>
<td>-1</td>
<td>3</td>
<td>938</td>
<td>1</td>
<td>0.3173</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>5</td>
<td>-1</td>
<td>4</td>
<td>47</td>
<td>1</td>
<td>0.0159</td>
<td>1</td>
</tr>
</tbody>
</table>

The rules in Table 4.6 are applied to the inference engine in two different ways. The first way is simply by applying all certain rules. The second way is by applying the rules with the maximum support value. So, for each set of rules that predict the same cluster,
only the rule with the maximum support is applied to the inference engine. For example, if there is more than one rule that predicts cluster no. 1, then the rule with the maximum support value will be used and the other rules will be disregarded. The advantage of the maximum support method is that it describes a system with less number of rules.

4.5 Classifier accuracy estimator

We apply the holdout classifier accuracy estimator [51] to estimate the accuracy of the different classifiers used. As shown in Figure 4.7, the examples generated from the clustering process were randomly partitioned into two independent sets: \( \alpha \) percent of the data were used as the training set, and the rest, i.e., \( (1-\alpha) \) percent of the data, were used as a testing set. The training set was used to generate rules either by the centroid method or by the BLEM2 classifier. The testing set along with the generated rules were applied to an inference engine to predict the classes for the testing sets based on the generated rules. The overall average accuracy result is the percentage of correctly predicted classes out of the overall testing set.
To illustrate how the overall average accuracy is calculated consider the following two rules and the testing examples in Table 4.7

\[
\text{if } 2^{\text{nd}} \text{ Page} = p_{13} \text{ then Cluster} = C_1
\]

\[
\text{if } 3^{\text{rd}} \text{ Page} = p_{85} \text{ then Cluster} = C_2
\]

Table 4.7 Inference engine testing examples

<table>
<thead>
<tr>
<th>Example No.</th>
<th>1\textsuperscript{st} Page</th>
<th>2\textsuperscript{nd} Page</th>
<th>3\textsuperscript{rd} Page</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$p_{45}$</td>
<td>$p_{13}$</td>
<td>$p_{50}$</td>
<td>$C_1$</td>
</tr>
<tr>
<td>2</td>
<td>$p_{45}$</td>
<td>$p_{13}$</td>
<td>$p_{66}$</td>
<td>$C_1$</td>
</tr>
<tr>
<td>3</td>
<td>$p_{45}$</td>
<td>$p_{12}$</td>
<td>$p_{85}$</td>
<td>$C_2$</td>
</tr>
<tr>
<td>4</td>
<td>$p_{45}$</td>
<td>$p_9$</td>
<td>$p_{85}$</td>
<td>$C_2$</td>
</tr>
<tr>
<td>5</td>
<td>$p_{45}$</td>
<td>$p_{12}$</td>
<td>$p_{85}$</td>
<td>$C_2$</td>
</tr>
<tr>
<td>6</td>
<td>$p_{45}$</td>
<td>$p_{13}$</td>
<td>$p_{33}$</td>
<td>$C_3$</td>
</tr>
<tr>
<td>7</td>
<td>$p_{45}$</td>
<td>$p_{10}$</td>
<td>$p_{85}$</td>
<td>$C_4$</td>
</tr>
</tbody>
</table>

The inference engine predicts the classes of the examples in Table 4.7 using the two rules presented earlier. The predicted classes are compared to the given classes in Table
Table 4.8 shows the inference engine clustering prediction along with classes given in Table 4.7. Because we have 5 matches out of 7 examples, the overall average accuracy for the classifier that generated the rules is \( \frac{5}{7} = 0.71 \).

<table>
<thead>
<tr>
<th>Example No.</th>
<th>Cluster from examples</th>
<th>Cluster prediction from the inference engine</th>
<th>Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( C_1 )</td>
<td>( C_1 )</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>( C_1 )</td>
<td>( C_1 )</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>( C_3 )</td>
<td>( C_1 )</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>( C_4 )</td>
<td>( C_2 )</td>
<td>0</td>
</tr>
</tbody>
</table>

### 4.6 Experimental results

In this section, we present the experimental results for the Web Usage Mining (WUM) clustering and learning algorithms described in this chapter. First, we present the choice of the termination condition for the two clustering algorithms. Next, we present the accuracy of the prediction sessions’ clusters using the rules generated using different classifiers described in Section 4.4. Finally, we present the experimental results for using the rules for describing the clusters by presenting the average cluster description length using different classifiers.

#### 4.6.1 Choosing the clustering termination conditions

The clustering algorithm termination condition depends on the number of clusters or, more precisely, on the percentage of the number of the clusters from the initial number of clusters. The first termination condition we chose is when all sessions in the
same cluster have the exact same sequence. This occurs at the first iteration when the percentage number of clusters is 100%. For the second termination condition, we choose the session with the shortest length to determine the iteration at which we stop the clustering. Then, we find the percentage of the number of clusters at that point. For the rest of the session length groups, we stop at the point where the percentage of the number of the clusters is closest to the percentage of the number of clusters for the shortest session length. The reason we choose the session with the shortest length is because it is most sensitive session to the threshold. To illustrate this, consider the following pair of sessions, where the first pair of sessions is of length 3:

\[ p_0, p_1, p_2 \\
  p_3, p_4, p_5, \]

and the second pair of sessions is of length 10:

\[ p_0, p_1, p_2, p_0, p_1, p_2, p_0, p_1, p_2, p_0 \\
  p_3, p_4, p_5, p_1, p_5, p_6, p_7, p_8, p_9, p_{10}. \]

For the previous two pairs of sessions, if we have a threshold of 3 of differences, the first pair of sessions, which have the length of 3, will be a 100% match; whereas, the second pair of sessions, which have the length of 10, will be a 30% match according to the distance equation 3.10 presented in Chapter 3. Even though the two pair of sessions have a complete mismatch, the threshold value of 3 caused the session of length 3 to give a 100% match, and the session with length 10 to give a reasonable difference of 30%.

In our experiment, a session with length 3 is considered to be the session with the shortest length. As shown in Figure 4.8, for session length 3, the percentage of the number clusters dropped to a value close to zero after 8 iterations. So our choice for the
termination condition was between iterations 1 and 8. Iteration 1 was chosen for the first set of the experiment. From the rest of the iterations, we choose iteration 2 since the rest of the iterations show a small percentage of number of clusters. At iteration 2, the percentage of number of clusters was 15.69%. For other session lengths, we stop iterations where the percentage of number of clusters has the closest value to 15.69%.

![Figure 4.8 Percentage of the number of the clusters from the initial number of clusters for different session length groups at different iterations](image)

4.6.2 Classifier prediction accuracy results by rules generated from examples using the hierarchal clustering algorithm

In this subsection, we present the classifier prediction accuracy results where the rules are generated from the examples using the hierarchal clustering algorithm described in Section 4.3.3. The first test set is performed on the examples generated using the clustering algorithm, where the termination condition is 100% of number of clusters. Figure 4.9 shows the average accuracy where BLEM2 (all) refers to all rules generated using BLEM2, BLEM2 (max) refers to BLEM2 rules with the maximum support only,
and centroid refers to rules generated using the centroid method. The accuracy was constant with a value of 1 for all three different classifiers.

![Figure 4.9 Average accuracy for different session lengths at the 100% number of clusters using examples from the hierarchical clustering algorithm](image)

The second test set was performed on the clusters generated when the number of clusters was 15.6%. Since the clustering results depend on the seed starting point for clustering, the experiments were repeated five times for each session length group. The overall average accuracy results are shown in Figure 4.10. The results clearly show that the average accuracy using BLME2 is better than the centroid method.
4.6.3 Classifier prediction accuracy results by rules generated from examples using equivalence classes clustering algorithm

In this subsection, we present the classifier prediction accuracy results where the rules were generated using the examples from the equivalence classes clustering algorithm described in Section 4.3.4. The first test set was performed on the examples generated from the clustering algorithm where the termination condition was at 100% of number of clusters. The results were same results, for the hierarchal clustering algorithm accuracy, shown in Figure 4.9—where the accuracy was a constant value of 1 for all three different classifiers.

The second test set was performed using the same termination condition used in the hierarchical clustering algorithm, which is 15.6% of number of clusters. Since the clustering results are independent of the seed starting point, the experiments were performed only once and the accuracy results are shown in Figure 4.11. As the case for the hierarchal clustering, the results show that the average accuracy using BLME2 is better than the centroid method.
4.6.4 Cluster description results

As described in Section 4.4, rules learned using the centroid method and BLEM2 methods are used to describe the clustering results. The cluster description length is defined as the number of conditions in the if part of the if-statement that represents the rule. For example, if we have the following two rules that describe clusters $C_1$ and $C_2$, respectively

\[
\text{If } 2^{nd} \text{ Page} = p_{13} \text{ and } 3^{rd} \text{ Page} = p_{14} \text{ then Cluster} = C_1
\]

\[
\text{If } 1^{st} \text{ Page} = p_{45} \text{ and } 2^{nd} \text{ Page} = p_{33} \text{ and } 3^{rd} \text{ Page} = p_{85} \text{ then Cluster} = C_2,
\]

then clusters $C_1$’s description length is 2 and cluster $C_2$ ’s description length is 3. Thus, the average description length is $\frac{2 + 3}{2} = 2.5$. Figure 4.12 shows the cluster description length using the different classifiers. The BLEM2 classifier shows a short constant cluster description length over different session lengths, while the centroid method shows a linear increasing cluster description length.
4.7 Results incorporation

Incorporating the results to enhance the structure of the website is done in three steps. The first step is to identify the most common tasks. The second step is to find how many clicks it takes to finish each task. The last step is to present suggestions for enhancing the structure of the website so that the common tasks can be done easier and faster.

4.7.1 Identifying the most common tasks

Each web session cluster represents one task and the number of the sessions in the cluster reflects how common the task is. For example, Figure 4.13 shows the seven most common tasks performed on the University of Akron registrar website. The task description was identified by the cluster description described in Section 4.4.
4.7.2 Finding how many clicks needed to finish each task

We assume that each page in the session represents one click for the user to move from one page to another, so the total number of clicks needed to finish each task is the same as the sequence length. Figure 4.13 shows the distribution of different number of clicks for “Class search detail” task. From the figure, it can be concluded that in 61% of the time the “Class search detail” was finished in 5 clicks, while in 24% of the time the task was finished in 3 clicks. From the cluster centroid, the following page sequence was clicked to finish the task in 5 clicks

\(\langle\text{signon, BeforeClassSearch, ClassSearch, BigClassSearch Result, ClassSearchDetail}\rangle\)

When the task was completed in 3 clicks the following page sequence was clicked

\(\langle\text{signon, BigClassSearch Result, ClassSearchDetail}\rangle\)
4.7.3 Presenting suggestions to enhance the website structure

By studying how different tasks are completed, recommendations can be made to change the website structure to permit common tasks to be completed in a shorter time and with lesser number of clicks. For example, for the “Class search detail” task example presented earlier, it can be seen from the clusters centroids that the users, who finished the task in 5 clicks, went through regular “class search” first before going to the “class search detail”; whereas, the users, who finished the task in 3 clicks, directly checked the “class search detail”. Conclusions can be made to the website engineer to have a shortcut for the “class search detail” on the homepage so users can directly access it rather being forced to go through the “class search” first.

4.8 Results discussion

The choice of a common termination condition for all session length groups must be based on the session with the shortest length. As shown in Figure 4.8, the session with length 3 was used to determine the common termination condition by founding the percentage of number of sessions at the second iteration for session length 3.
Figure 4.9 shows that when sessions in the same cluster have the same exact page sequence, the prediction accuracy is 1 for all different classification methods. When the threshold is increased, Figure 4.10 and Figure 4.11 show that the rough set based BLEM2 rules predict the classes for sessions more accurately. This is for rules generated using examples from hierarchal and equivalence classes clustering algorithms. Results shown in Figure 4.12 illustrate not only do BLEM2 rules predict the cluster description more accurately, but it also presents a shorter description for the clusters. Figure 4.12 shows that the cluster description using the centroid method is linear and increases as the session length increases, while the cluster description based on rules learned using BLEM2 is almost constant and has a length around 2.

Figures 4.13 and 4.14 show how the clustering and learning results may be used to give insightful information about the website, like what are the most common tasks, and how these tasks are commonly achieved. Finally, Section 4.7.3 shows how this information can be used in enhancing the website structure to make the users’ task achieved faster and easier.

4.9 Summary and conclusion

In this chapter, we presented two different clustering algorithms to generate examples that can be used by different classifiers. We used both the centroid and BLEM2 classifiers to learn rules from the examples generated using the clustering algorithms. We applied the holdout classifier accuracy estimator to measure the accuracy of the classifiers. Rules generated by BLEM2 show a better cluster prediction and shorter cluster description.
The rules generated by different classifiers were used to present a deep conceptual understanding of the usage behavior of the website, which can be used by the website engineer to evaluate and to enhance the website structure and predict future users’ browsing behavior to better assist users in their future browsing experiences.

The work presented in this chapter—including generating examples, learning rules, and testing the results—can be applied to sequence clustering methods in other fields, such as bioinformatics, which is the area of analyzing genomic research data.
CHAPTER V
SYSTEM IMPLEMENTATION

5.1 Introduction

In this chapter, we present the implementation of the web usage mining system presented in the previous chapters. As shown in Figure 5.1, the implementation is divided into four modules: data preparation, session identification, clustering process, and result presentation and evaluation. The data preparation module performs data filtering and user identification. The session identification module performs session identification and further data filtering. The clustering process module generates the dissimilarity matrix and performs different clustering algorithms; including hierarchal and equivalence classes clustering algorithms. The result presentation and evaluation module performs the learning process along with accuracy estimation for learning results.

The reset of the chapter is organized as follows. In Section 2, we present the implementation of the data preparation module. In Section 3, we present the implementation of the session identification module. In Section 4, we present the implementation of the clustering process module. In Section 5, we present the implementation of the result presentation and evaluation module. Finally, in Section 6, we present a summary.
5.2 Data preparation module

The data preparation module performs both data filtering and user identification.

The implementation is done using MS SQL server. Figure 5.2 shows the entity relation
model (EM) for the database design. The Web_Records table contains all the raw data collected from the web server. The filtered records are then stored in the Web_Log table. The Open_Users and Users tables are used by the active user-based user identification script shown in Figure 2.7. The final results of the user identification process are stored in the Users table.

![Entity relation model for data preparation](image)

**Figure 5.2 Entity relation model for data preparation**

5.3 Session identification module

As shown in the use case diagram in Figure 5.3, the session identification module allows the user to perform different tasks, such as loading user records, to perform session identification, to perform further data filtering, and to export the results to different platforms.
Figure 5.3 Use case diagram for session identification

Figure 5.4 shows the session identification module user interface. The first step in using the program is to load the sequence file, which is the output from the user identification step. Next, the user is asked to load the page lookup file which matches the page numbers used in the sequence file with its physical name. The page lookup file also indicates if the page is a housekeeping page or not. Several filtering options are available:

- Remove housekeeping pages: this option removes the housekeeping pages as identified by the user in the page lookup file. The user can setup these files by pressing the “Setup House Keeping Pages” button.
- Remove redundant pages: this option removes the redundant pages that resulted from removing the housekeeping pages.
- Session identification based on break pages: this option splits users’ records into one or more sessions based on break pages. To do this, the user needs to enter the page breaks. Then, the program will run the algorithm described in Section 2.3.3.
For our case we chose the *sign-in* page as a break page and the sessions were identified based on that.

- Specific session length range: this option filters sessions that have a specific number of records.

After these operations are completed the results can be exported into different formats:

- Space delimited: this format is a general purpose format that can be read by many learning tools including ours.

- Weka [52]: this format can be read by the popular open source machine learning program Weka. Weka, as many other machine learning programs, requires a fixed sequence length, so this option can’t be used unless the data is filtered to fixed length.

- Result statistic: this format gives statistical results about the distribution of the length of the sessions after filtering.
5.4 Clustering process module

In the clustering module, users first prepare the session data for clustering by generating the dissimilarity matrix. Hierarchical and equivalence class clustering algorithms can then be applied to the sessions using the generated dissimilarity matrix. Finally, users can pick clusters at a certain level of threshold. Figure 5.5 shows the use case diagram for the clustering process module.
Figure 5.5 Use case diagram for clustering process module

Figure 5.6 shows the UML diagram for the clustering process module. The multiplicity shows that the dissimilarity matrix is generated based on the sessions. Each cluster has one or more sessions, and each session belongs to one cluster only. Finally, the clusters are generated using the hierarchal class. Each cluster consists of one or several cluster levels.

Figure 5.6 UML diagram for the clustering process

Figure 5.7 shows the first sequence diagram in the clustering process. This diagram shows how the dissimilarity matrix is generated by passing different messages between Session, Dissimilarity Matrix and Distance classes.
Figure 5.7 Sequence diagram for generating dissimilarity matrix

Figure 5.8 shows the second sequence diagram in the clustering process. This diagram shows how the clusters are found by passing different messages between Cluster, Hierarchal and Dissimilarity Matrix classes.

Figure 5.8 Sequence diagram finding clusters
Figure 5.9 shows the clustering module user interface. The user starts by loading the session file prepared earlier by the session identification process. Then, the dissimilarity matrix can either be generated, by choosing the “Generate Dissimilarity Matrix” button, or loaded directly from a text file, using the “Load Similarity Matrix” button. Once both the session and the dissimilarity matrix files are ready, the user can perform clustering process by choosing “Run Clustering” button. The user can then export the clustering result at any level, by providing the clustering level in the “Clustering Level” space, and then by clicking “Export Clustering Results at Certain Level” button.
5.5 Results presentation and evaluation module

Figure 5.10 shows the dataflow diagram for the results presentation and evaluation module. The figure shows the programs used at different steps in the learning and evaluation process. Split.java is used to split the clusters into two dependent sets. Raff2Lem.ext is used for exporting the result to BLEM2 format and then using Lers7.exe for the learning process to generate the first set of rules using BLEM2. MaxSupport.java
is used in filtering out rules that have maximum support. FindCentroid.java is used in learning rules using the centroid method. OpMVClassifierCF3.tcl is the inference engine that tests the accuracy of the classifiers. Source code for Split.java, FindCentroid.java, MaxSupport.java and OpMVClassifierCF3.tcl are available upon request.
5.6 Summary

In this chapter, we presented the implementation of the web usage mining system presented in this dissertation. The system implantation was accomplished using mixture
of different programming environments such as SQL, Java and TCL. The source code for
the implementation is available upon request.
CHAPTER VI
SUMMARY AND CONCLUSIONS

In this work, we presented a complete Web Usage Mining (WUM) system using data mining techniques and a rough set learning approach. The system architecture covered the major parts of the WUM system including data preprocessing, data cleaning and filtering, session comparison, clustering analysis, and results presentation and results incorporation. The goal of this system is to give a deep conceptual understanding of the usage behavior of a website. This conceptual understanding can be used by the website engineer to evaluate and to enhance the website structure to better assist users in their future browsing experiences.

In the data preprocessing phase, we presented new techniques for preprocessing web log data including identifying unique users and sessions. We developed a fast active user-based user identification algorithm which has a time complexity of $O(n)$. For session identification we presented an ontology-based session identification algorithm that uses the website structure to identify users’ sessions. We showed that the user identification algorithm depends on three parameters: number of records per user, web log records recording rate on the web log, and the maximum inactive time for users. Table 6.1 shows the mathematical models along with correlation coefficients for the three website parameters on which our active user-based user identification algorithm depends. These models can be used in simulating future website usage activity.
Table 6.1 Mathematical models for three website parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mathematical model</th>
<th>Correlation coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Records per user probability</td>
<td>Power fit</td>
<td>0.41</td>
</tr>
<tr>
<td>User navigation time probability</td>
<td>Reciprocal quadratic</td>
<td>0.92</td>
</tr>
<tr>
<td>Records per second probability</td>
<td>Geometric fit</td>
<td>1.00</td>
</tr>
</tbody>
</table>

In the session comparison phase, we presented a new multidimensional session comparison method (MSCM), which is based on dynamic programming. Unlike other algorithms, MSCM takes into consideration other dimensions in the session, such as the time spent on the page and the total session length. The algorithm provided more accurate results than other known algorithms in comparing web sessions, such as Sequence Alignment Method (SAM), Multidimensional Sequence Alignment Method (MDSAM) and Path Feature Space. The output of the MSCM is presented in the form of dissimilarity matrix, which can be used by different clustering techniques, such as hierarchal, k-mean and equivalence classes clustering algorithms.

In the clustering phase, we presented two clustering algorithms. The first is a hierarchal clustering algorithm and the other is an equivalence classes clustering algorithm. Unlike other clustering algorithms, the equivalence classes clustering algorithm does not depend on the seed starting point of the clustering process. So, we didn’t have to repeat the clustering process several times and to take the average; rather, it was sufficient to perform the clustering process once. We, also, presented a new method for choosing a common termination condition for clustering algorithms in the process of clustering different session length groups. The new method showed that the shortest session length must be used to determine the termination condition for other session length groups.
In the learning phase, the clustering results, which were presented in the form of examples, were used by two classifiers to generate rules. These rules were used in predicting the clusters for prospective users and to describe the cluster itself. We presented two classification approaches: the centroid approach and the rough set approach BLEM2. The accuracy for predicting the clusters for prospective sessions were measured using the holdout accuracy estimator method. The results showed that the rough set approach, BLEM2, is more accurate in predicting prospective sessions’ clusters. For the cluster description, we based our measure on the length of the description. The rough set approach BLEM2 showed shorter description length for clusters. In summary, the rules generated using the rough set approach BLEM2 better predict and describe web sessions’ clusters.

In the results incorporation phase, we used the clustering results along with the learned rules to present a deep conceptual description for a website usage. We presented the most common tasks that were performed on the website. In addition, we presented what a navigation path was most common used to complete these tasks. We showed how the clustering and learning results can be used in presenting suggestions to the website designer to enhance the website structure to better assist users in their future browsing experiences.

WUM research is an emerging field and there remains much to be learned from the interaction between users and different websites. Future work needs to be done to automate the process of the WUM. This can be carried out by incorporating the generated rules with the web log and clustering users into different clusters on the fly. Additional
work can also be done by dynamically adjusting the website structure according the WUM results.

Web sessions are a special case of string sequences, so as a future work the techniques presented in this dissertation— in particular the multi-dimensional sequence comparison algorithm, the two clustering algorithms, and the learning approaches— can be applied to other sequence comparison research areas, such as bioinformatics, which is the area of analyzing genomic research data.
REFERENCES


