An Empirical Study Of Web Usage Mining Techniques

Kwun-Keung, Ng

A Major Report

in

The Department

of

Computer Science

Presented in Partial Fulfilment of the Requirements

for the Degree of Master of Computer Science at

Concordia University

Montreal, Quebec, Canada

September 2002

0-612-72942-7

ABSTRACT

An Empirical Study Of Web Usage Mining Techniques

Kwun-Keung, Ng

Most of the existing web sites organize their content in a hierarchical manner. This organization may not be clear to the visitors because each visitor may have their own expected organization. For instance, it is often unclear to a visitor where a specific document is located. Usage knowledge, discovered from web usage mining, on the way visitors navigate in a web site could prevent disorientation, help the web site owner in designing the web site, provide efficient access between highly correlated object, and make better marketing decisions such as putting advertisements in proper places.

In this report, we will give an overview on the web usage mining process with special emphasis on presenting two data mining techniques: association rules and path traversal pattern discovery. We introduced the notion of *context awareness* web usage mining which is a constraint pushed into these data mining techniques. As well, we will present our design and implementation of a web usage mining system call WUM. Finally, we will show the experimentation of using our WUM to mine for usage knowledge for a Computer Science department's web site.

Acknowledgments

Thanks to my supervisor, Dr. Shiri Nematollah, who guided me through this project and gave me encouragement, assistance and valuable opinions. Also, I would like to thank my wife, Chiu, Siu-Fung, for her continued encouragement and support. Finally, thanks to the department of Computer Science of Concordia University for making their access logs available for this project.

Table of Contents

List of Figures

1. Introduction

1.1 The Problem

More and more organizations rely on the Internet and the World Wide Web to conduct business or share information. The problem of designing a web site is how to organize the information on a page and the hyperlinks between pages to serve users "properly". In order to design a "good" web site, a web designer has to know how the visitors may use the web site. In a small web site with less than 100 pages, the web designer can use common sense and simple statistics provided by *Web Log Analysis Tool* [11] to predict user's navigation behaviour. However, for large web sites having over 1000 pages, the size and complexity increase, therefore it is inadequate to use common sense and simple statistics to determine the usage of such web sites.

The Web Server hosting a web site generates the usage information of the web site automatically. This information is stored in the Web Server as a log file. The content of this log file is commonly referred to as *user access logs*. Despite the evolving technology and research in web mining, it is still not clear how to analyse user access logs and how to extract the usage of a web site from these user access logs.

1.2 Web Usage Mining

Web usage mining is the application of data mining techniques to large user access logs in order to extract usage patterns [6]. Web usage mining techniques have become important for a number of applications such as web site design, business and marketing decision support, personalization, usability studies, and network traffic analysis. Figure 1 shows the components of a Web usage mining process. Basically, this process consists of a pre-processing cleaning phase, which uses the raw user access logs from the server to generate *user session file* (also called user transactions). Then, depending on the particular interest, various data mining techniques maybe applied on these session files. Finally, the results obtained from applying data mining techniques to these session files are analyzed to generate rules, patterns, and statistics. There are three major steps involved in the web usage mining process, which give rise to the following questions.

First, how to pre-process the raw data in order to obtain an accurate picture of how a site is being used? Second, what data mining techniques to apply in order to discover usage patterns? Last, how to filter out the results of the various data mining algorithms in order to present only the rules and patterns that are "potentially interesting" in a given context?

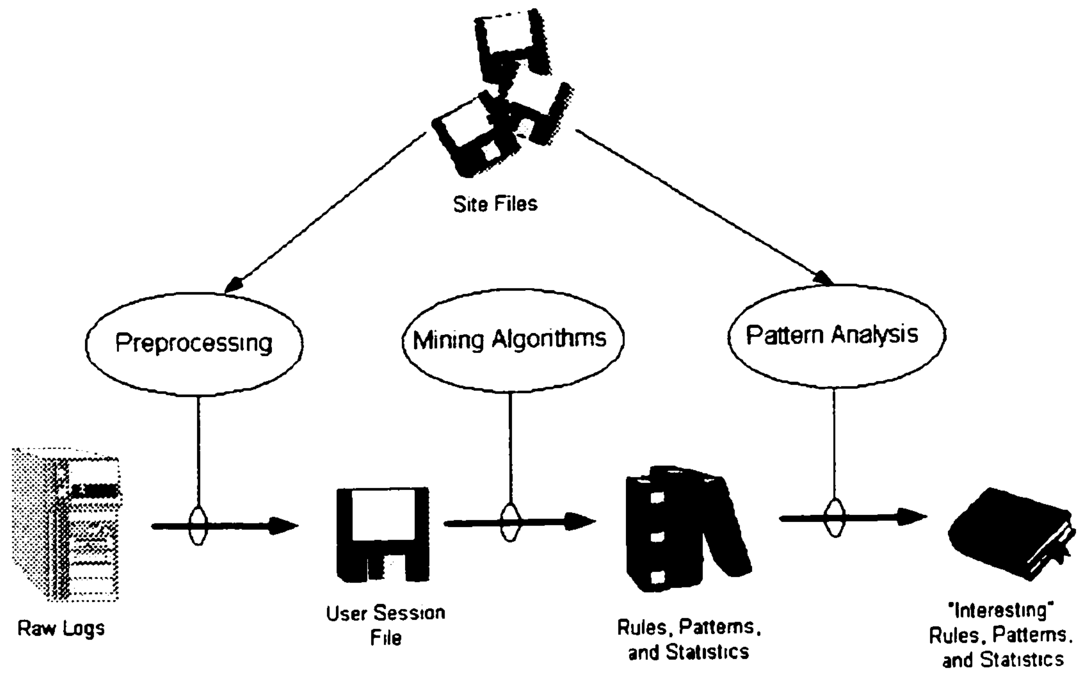


Figure 1. High Level Web Usage Mining Process [8]

The rest of this report is organized as follows: Section 2 discusses the pre-processing phase. Section 3 explains four data mining techniques commonly used for web usage mining with emphasis on the association rule and path traversal mining techniques. In section 4, we present our design and implementation of a system, which we call WUM for studying and experimenting with various Web Usage Mining (WUM) techniques. In this system, we implemented the algorithms for association rule mining, path traversal mining, and an extension of path traversal mining to mine for usage knowledge. In section 5, we present a case study of using WUM to mine for usage knowledge in the web site of a computer science department. The experimental results and our analyses are reported in section 6. Finally, section 7 includes some concluding remarks and future works.

2 Pre-processing

The amount of access log records generated on the web server for an average accessed web site today, about 1000 hits per hour, can easily reach tens of megabytes per day, which would cause the analysis of log records to be rather slow and inefficient. These access log records are stored in a log file, which is essentially a sequential file. Some of the access logs are meaningless for the majority of user access pattern discovery techniques, because they do not provide an "appropriate" picture of the access patterns of the web site users. This will be discussed in detail in section 2.2. Therefore, the data should be pre-processed before mining for usage patterns.

2.1 Web Data

In a web mining process, data might be collected from different locations such as the server-side, client-side, proxy servers, or from the organization's database. Each of these data may contain different kinds of information, classified as follows:

1. *Content*: The data in the web pages such as texts, images, videos, sounds are intended to be transported to the users.
2. *Structure*: The data that describes the structure and the organization of the information through internal tags or hyper-links.
3. *Usage*: The data that describes the usage pattern of the web pages. This data is stored as access logs in a log file on the Web Server.
4. *User Profile*: Demographic information of the users derived from user registration.

In our study that will be presented in section 4, we used the usage data (user access logs) that is automatically gathered by the web server. The web server stores these data as log record in a log file. When a user request is sent to the Web Server, it will add an entry into the log file. For example, when a user requests the page www.cs.concordia.ca through his/her web browser, the web server will record an entry in the log file for this request. These entries reflect the access of the web site by various users. The log file can be stored in various formats depending on the Web Server. Figure 2 shows a typical example of the structure and a sample of user access logs generated by a Web Server. An explanation of the structure and the values in this figure is as follows:

1. *IP Address:* The client IP address.
2. *Time:* The time of the request.
3. *Method URL Protocol:* The method is the request method (GET, POST, HEAD), the URL is the page name, and the transmission protocol is either HTTP/1.0 or HTTP /1.1.
4. *Status:* The status code returned by the web server as a response.
5. *Size:* Number of bytes transmitted for this request.
6. *Agent:* The client application used to send the request.

However, the information in this log file cannot fully record the whole access patterns of the users because of the presence of various levels of caching in the Web environment. For example, if user A asks for the page www.cs.concordia.ca, and his/her browser has already cached this page, then the browser will get this page from the cache instead of requesting it from the Web Server. In other words, no access log entry will be created for this particular access.

IP Address	**Time**	**Method URL Protocol (The URL is the page name)**	**Status**	**Size (bytes)**	**Agent**
132.205.44.148	[20/Feb/2002:01:05:30 - 0500]	"GET / logoconc.gif / HTTP/1.0"	200	2770	IE
130.191.40.70	[20/Feb/2002:01:06:10 - 0500]	"GET /~grad/mlhossa/Final_Slides.pdf HTTP/1.0"	404	100	Netscape
65.92.160.180	[20/Feb/2002:00:32:05 - 0500]	"GET /programs/ugrad/cs/comp249.shtmlHTTP/1.0"	200	6532	Mozilla
65.92.160.180	[20/Feb/2002:00:38:02 - 0500]	"GET /cgi-bin/Course.cgi?file=Schedule_Winter_2002&course=comp+249 HTTP/1.0"	200	7980	IE
65.92.160.180	[20/Feb/2002:00:41:04 - 0500]	"GET / / HTTP/1.0"	200	8100	IE
65.92.160.180	[20/Feb/2002:00:42:08 - 0500]	"GET / logoconc.gif / HTTP/1.0"	200	2770	IE

Figure 2. User access logs

2.2 Data Cleaning and Transaction Identification

The raw user access logs have to be cleaned and transformed into a table of sequences of pages accessed, which gives a clearer picture of the user accesses to the web site. A user's request to view a particular HTML page often results in generating several access log entries since graphics and scripts are downloaded in addition to the HTML file. Since HTTP connections are stateless, several requests to the Web Server are made in order to fulfill a single user request. For example, suppose a user requests the page current_students.shtml that contains four images. Then, five requests are sent to the web server that is hosting this page. The first request is to access the page current_students.shtml, and the other four requests are for the four images in this html page. Cleaning the user access logs consists of removing all the "irrelevant" access log entries that does not represent an *actual action of the user*. In the case of the above request, images are consider as non user initiated action, so the clean up process should remove the four access logs entries for the four images and leave the request for the page current_students.shtml - the only user initiated action.

2.2.1 Remove Non-user Initiated Requests

Elimination of non-user initiated access log entries can be accomplished by simply checking the suffix or extension of the page name as specified in [2]. All log entries for pages with extensions such as, gif, jpeg, GIF, JPEG, jpg, and JPG can be removed, since these are requests for images and users do not often initiate them. For example, the first entry in the log table in Figure 2 will be removed since it is an image with the URL logoconc.gif, which contains .gif extension.

2.2.2 Remove Unsuccessful Requests

In hypertext transport protocol (HTTP), status code is used to show the status of the requested page. For instance, if a request for a page fails, this will be indicated in the *Status code*, which is an attribute of the log entry. Access entries corresponding to failed requests could be easily identified and removed. This can be accomplished by checking the status code of the access log as specified in [2]. For example, a request for a page that no longer exists on the server will create an access log with a status code of 404, indicating an error. Any request for a page that causes a server error will create a log entry with a status code of 5XX where XX is some two digits number. A successful request will create a log with a status code of 2XX. For web usage mining, we are only interested in requests that successfully find the requested pages. Therefore, we filter out all access logs that have a status code different from 2XX. For example, the second entry in the log table in Figure 2 will be removed since its status code is 404 which is not of the form 2XX

2.2.3 User Transactions Identification

In addition to the above pre-processing methods, the user access logs have to be transformed into a form suitable for pattern discovery. The common data often used by pattern discovery algorithms is the identification of user transactions from these logs. A user transaction is a sequence of pages accessed by a user across a web site in a specific period of time. The goal of user transaction identification is to group meaningful clusters of pages accessed by each user. Such transactions will be used later on for pattern discovery. For details of the proposed methods for transactions identification, interested readers are referred to [8]. As mentioned earlier, due to caching on the client side, it is important to realize that the access log may not contain all the pages accessed by a user, and hence a user transaction in this sense may be incomplete.

3 Applying Data Mining Techniques

Data mining is a process of inferring knowledge from huge amount of data. It includes among others, techniques for discovering association rules, clustering, classification, and sequence analysis. Once user transactions have been identified as discussed in section 2.2, there are different kinds of data mining techniques that can be performed depending on the needs. However, our intent in this report is not to review all the proposed techniques. We will discuss the following four data mining techniques often used to discover usage patterns.

- Association Rule [1]
- Classification [10]
- Path Traversal Mining [4]
- Sequential Patterns [2]

3.1 Association Rules Mining

Association rule discovery techniques [1] were proposed for mining in databases of user transactions where each user transaction consists of a set of pages accessed by a client over a period of time. Let $I = \{ i_1, i_2, i_3, \ldots, i_k \}$ be the set of pages sand $T = \{ t_1, t_2, t_3, \ldots, t_k \}$ be the set of user transactions. An association rule is a statement of the form X=>Y, where $X \subseteq I$, $Y \subseteq I$ are disjoint sets of items called *itemsets*. From the set of user transactions, we want to find all association rules with confidence and support greater than or equal to a specified pair of thresholds confidence (C) and support (S). A confidence C indicates that C percentage of the transactions that contain X also contain Y. Support S is a measure based on the number of occurrences in all the user transactions.

To be more precise, S is the ratio of the number of transaction that contains items in X U Y and the number of transactions.

The following examples show samples of knowledge that can be discovered using the association rule mining techniques.

1. 40% of clients who accessed the Web page with URL /company/products/product1.html, also accessed /company/products/product2.htm
2. 30% of clients who accessed the page /company/announcements/specialoffer.html, placed an online order in /company/products/product1.

Since there are a large number of user transactions, most association rule discovery techniques try to identify and prune data that are less relevant according to a certain *support*. Discovery of such association rules can help in the development of more effective marketing strategies. In addition, association rules discovered can provide hint as how we may organize a web page and decide the hierarchical structure of pages in the web site. For example, if we discover that 80% of the clients accessing /company/products and /company/products/file1.html also accessed /company/products/file2.html, but only 30% of those who accessed /company/products also accessed /company/products/file2.html, then it is likely that some information in file1.html leads clients to access file2.html. This correlation might suggest moving this information to a higher level (e.g., /company/products) to increase and facilitate access to file2.html.

3.2 Classification Methods

Discovering classification rules [10] allows one to develop a profile of pages belonging to a particular group according to their common attributes. This profile can then be used to classify new pages that are added to the web site. In web mining, classification techniques allow one to develop a profile for users who access particular pages based on demographic information available on those users, or based on their access patterns. The following examples show samples of knowledge that can be discovered using classification algorithms for pages.

1. Clients from state or government agencies who visit the site tend to be interested in the page /company/products/product1.html.
2. 50% of clients who placed an online order in /company/products/product2, were in the 20-25 age group and lived on the West Coast.

User registration and online survey, if available, can be used to get profile and demographic information on clients.

3.3 Path Traversal Mining Techniques

Path traversal mining could be used to determine most frequently visited paths in a web site. The following examples illustrate the kinds of knowledge that can be discovered using path traversal mining techniques.

1. 70% of clients who accessed /company/products/file2.html did so by starting at page /company and then proceeding through the pages /company/whatsnew, /company/products, and /company/products/file1.html. The example suggests that there is useful information in page /company/products/file2.html, but users tend to take an indirect route to the page.
2. 80% of clients who accessed the site started from the page /company/products. This example simply states that the majority of users are accessing the site through a page other than the main page (assumed to be /company in this example)

and it might be a good idea to include directory type information on this page if it is not there already.

3. 65% of clients left the site after four or less page references. This indicates that not many users browse further than four pages into the site. So, it would be better to ensure that important information is provided within the first four pages from the main page.

One technique to mine for the paths traversed by the users is to get the maximal forward references from the user transactions as described in [4]. Using the maximal forward references discovered, we can represent the paths traversed using graphs, which can be used to conveniently represent the hierarchy among web pages. To be more specific, such a graph can be used to represent the physical layout of a web site, with web pages as nodes, and hypertext links between the pages as directed edges. Other graphs could be defined based on the types of web pages with edges representing similarity between pages, or creating edges that give the number of users that browsed from one page to another [9].

3.4 Sequential Patterns Mining Techniques

The problem of discovering sequential patterns [2] is to find inter-transaction patterns such that the presence of a sequence of pages is followed by another page in user transactions ordered by time. The discovery of sequential patterns allows organizations to predict user visit patterns. It also helps in placement of advertisements. The following examples show the kinds of knowledge that can be discovered using the sequence pattern mining techniques.

1. 30% of clients who visited the page /company/products/, had done a search in Yahoo, within the past week.
2. 60% of clients who placed an online order in /company/products/product1.html, also placed an online order in /company1/products/product4 within 15 days.

The problem with sequence pattern mining is somewhat related to the problem of mining association rule. Association rules are rules about what items occur together within a transaction, which are intra-transaction patterns, unlike sequential patterns, which are sequences of items that occur together across transaction, which are inter-transaction patterns.

In the next section, we will present the design and implementation of a path traversal mining technique and an association rule mining technique. Since we are only interested in intra-transaction patterns, therefore, sequential pattern mining techniques are not implemented. Also, the classification methods are not implemented because we want to apply web usage mining for an anonymous user website, therefore user profile is not applicable.

4 Designs and Implementation

We have designed and implemented a Web Usage Mining system (WUM) to study mining techniques for sequence patterns and experimenting using the real user access logs from a CS department's web server. In this section, we present the architecture and the design of this system. We will also present implementation details.

4.1 The WUM Architecture

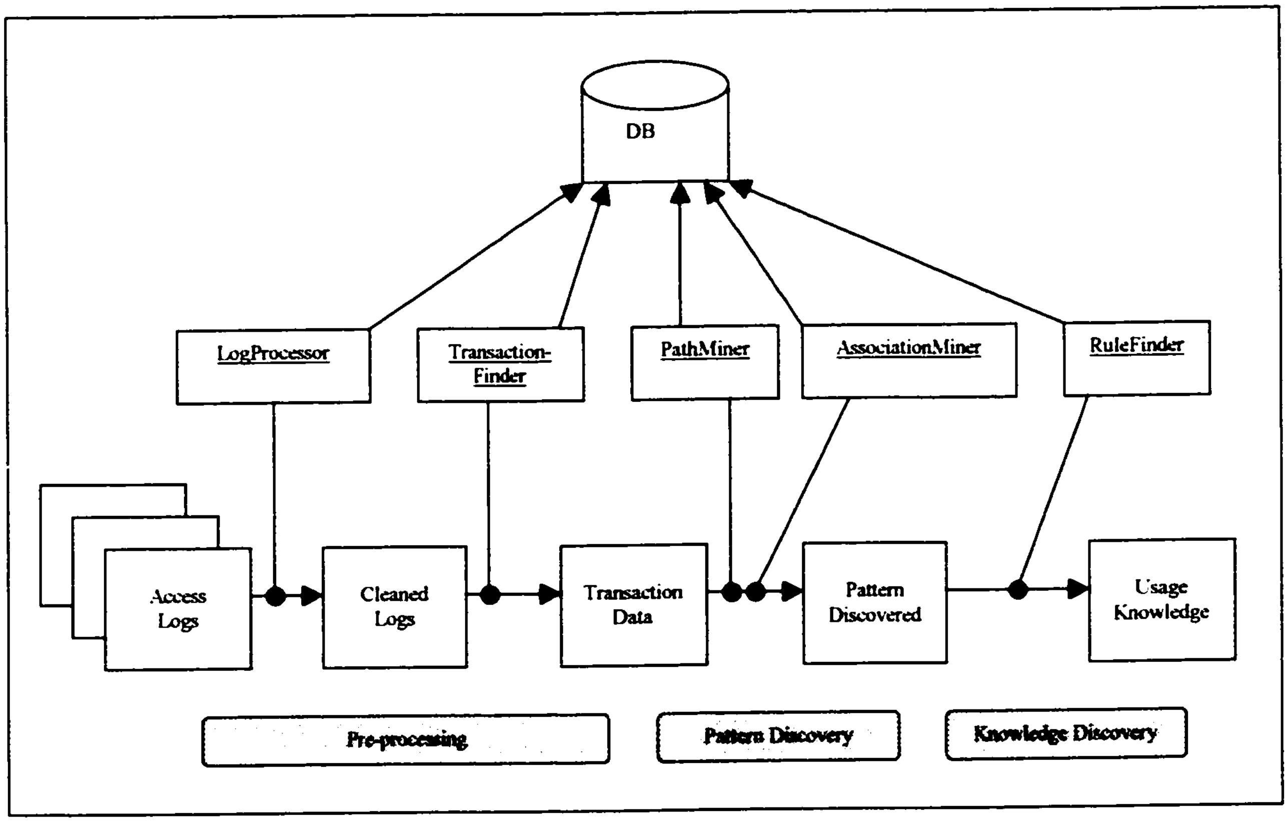


Figure 3. The WUM Architecture

The WUM system is composed of five main components:

1. *LogProcessor*: Reads the access logs, applies the cleaning techniques to clean the logs, and stores the cleaned logs into the database.
2. *TransactionFinder*: Identifies user transactions as defined in section 2.2.3 from the set of access logs in the database.

3. *PathMiner*: Applies a path mining technique on the identified user transactions.
4. *AssociationMiner*: Applies the association rule mining technique on the identified user transactions.
5. *RuleFinder*: From the pattern identified, it discovers some rules to be used later is analysis.

These components are used in different phases of the web usage mining process. First, the LogProcessor and TransactionIdentifier are used in the pre-processing phase. The PathMiner and the AssociationMiner components are used in the pattern discovery phase. The order of executing PathMiner and AssociationMiner is not important. Finally, the RuleFinder is used in the knowledge discovery phase.

4.2 Design of WUM

For this system, we used a top-down approach starting from identifying the use cases. A use case diagram describes the functionality of the system – what the system will do for the user in order to get some useful work done. We then defined the classes and the inter-relationship between those classes. Then, we created a collaboration diagram that describes the details of the flow of the sequence of events for our use cases. Figure 4 illustrates the use case of the WUM system.

4.2.1 WUM Use Case

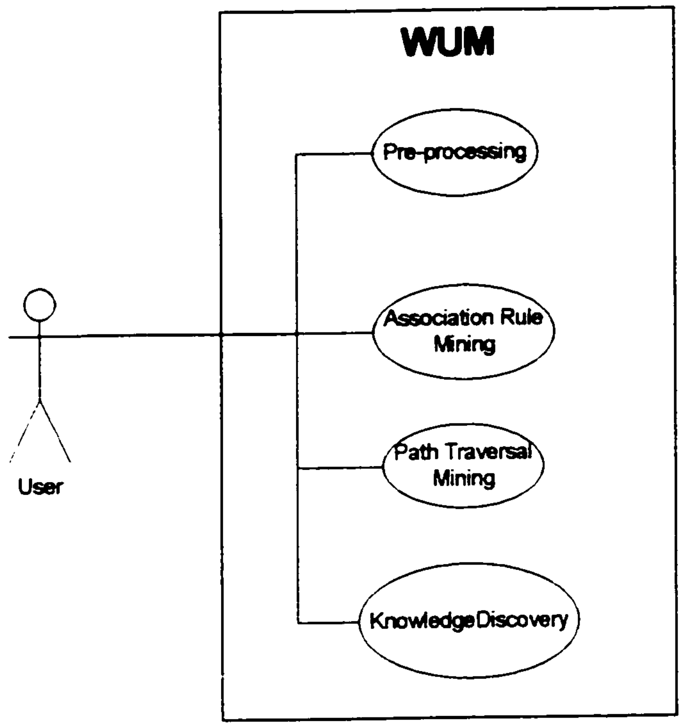


Figure 4. WUM Use Case

As shown in the figure, the WUM system has four use cases. The user will be able to request the system to perform pre-processing, Association rules mining, Path Traversal Mining, and knowledge discovery.

4.2.2 WUM Class Diagram

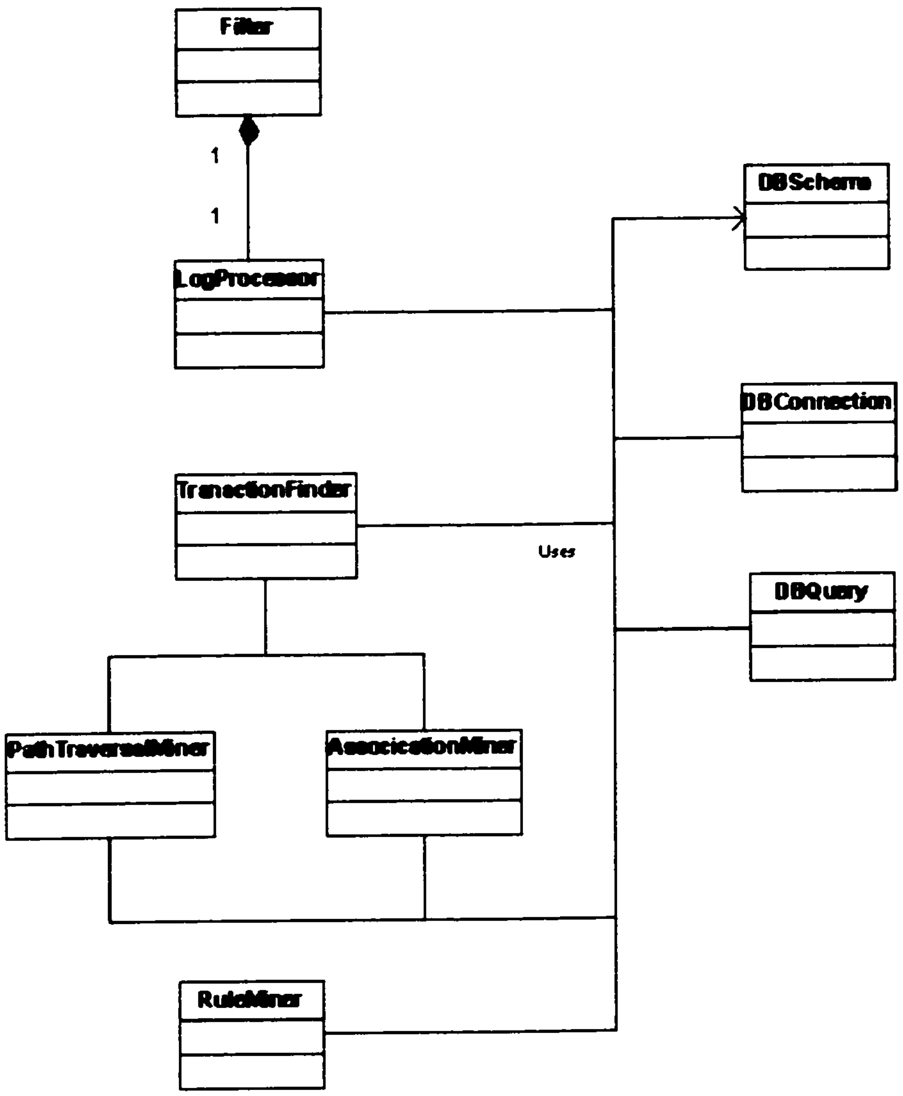


Figure 5. WUM Class Diagram

Figure 5 illustrates the classes used in the WUM system, as well as the relationship between them. A brief description of each class is as follows:

1. *DBSchema*: This class contains table and column names of the database used.
2. *DBConnection*: This class knows how to get a DB connection.
3. *DBQuery*: This class knows how to create different types of SQL statement.

4. *LogProcessor*: This class implements the data reading, data cleaning, and data insertion functionalities. It reads access logs from the specified access log file and it filters out all access logs that do not pass the filtering criteria. It then inserts all the access logs that satisfy the filtering criteria to the database.

5. *Filter*: Contains the logic to define filtering criteria and to determine if a page satisfies the filter criteria. To create a Filter, you will have to pass an array of strings and an array of integers to define the filtering criteria. The array of string is used to filter non-user initiated action. The array of integer is used to filter unsuccessful request.

6. *TransactionFinder*: This class is used to identify user transactions in the access log records in the database. When creating a TransactionFinder object, you can specify a session time. This session time will be use to define the duration of a user transaction. By default, the session time is 30 minutes.

7. *PathTraversalMiner*: This class implements the maximal forward reference algorithm with contextual constraint awareness. When there is no constraint specified, it executes like the original maximal forward reference algorithm. To create a PathTraversalMiner, you need to specify a minimum support and a constraint (if there is one).

8. *AssociationMiner*: This class implements the algorithm to generate the frequent itemset with 1 item and the Apriori algorithm. To create an AssociationMiner, you need to specify a minimum support.

9. *RuleMiner*: This class knows how to generate rules from association rule knowledge and path traversal knowledge. To create a RuleMiner, you need to specify a confidence value.

4.2.3 WUM Collaboration Diagram

WUM Collaboration diagram will detail the flow of the sequence of events for our use cases.

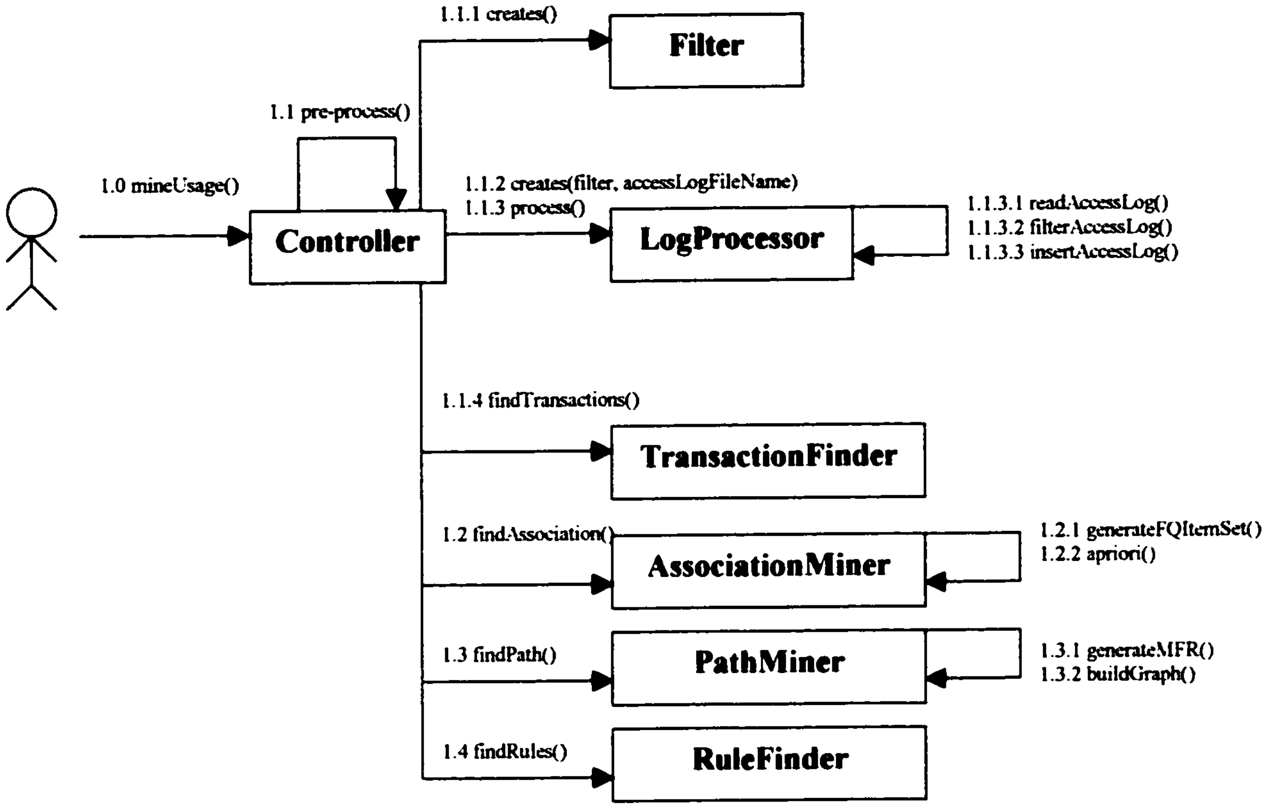


Figure 6. WUM Collaboration Diagram

The following are the details of each of the operation in the collaboration diagram:

1.0 The user requests WUM controller to discover usage knowledge.

1.1 The Controller initiates the pre-processing module.

1.1.1 The Controller creates the filter to be used for data cleaning.

1.1.2 The Controller creates a LogProcessor with the filter created in 1.1.1.

1.1.3 The Controller requests the LogProcessor to process access log file.

1.1.3.1 The LogProcessor reads the access log file.

1.1.3.2 The LogProcessor asks the Filter to filter the access logs.

1.1.3.3 The LogProcessor inserts the filtered access logs into the database.

1.1.4 The Controller requests the TransactionFinder to find the user transactions.

1.2 The Controller requests the AssociationMiner to discover association rules.

1.2.1 The AssociationMiner generates frequent itemsets for 1 item.

1.2.2 The AssoicationMiner runs the Apriori method to generate frequent itemset with 2 items and above.

1.3 The Controller requests the PathTraversalMiner to discover frequent accessed paths.

1.3.1 The PathTraversalMiner generates the maximal forward references.

1.3.2 The PathTraversalMiner builds the graphs with the generated maximal forward references.

1.4 The Controller requests the RuleFinder to find rules.

1.4.1 The RuleFinder finds associations rules from all the frequent itemsets.

1.4.2 The RuleFinder finds the frequently accessed paths from the generated graphs.

1.5 The Controller returns the rule generated to the user.

4.3 Implementation of WUM

The WUM is implemented using the Java programming language and the MS Access database in the Windows NT environment. WUM has 18 classes and 2378 lines of code. In this section, we will go into details of our implementation. In particular, we will discuss implementations of association rule mining technique and the path mining technique.

4.3.1 Association Rule Mining Technique Implementation

To generate association rules, we first find the set of frequent itemsets. Here are the two steps described in [1] to do this.

Step1. Find the frequent itemsets with 1 item. An item is consider to be frequent if no less than minimum support Smin% of all transactions contain X, that is, count(X) >= Smin % * number of transactions. Finding the frequent itemsets with 1 item consists of counting the number of times an item appears in all the user transactions. To compute this information, we examined all the transactions and kept a running count for each accessed page (e.g., all pages that were accessed). For each transaction T and for each item X, if T contains X, it increments the count of X. Finally, we return all the items that have at least Smin% support.

Step2: To discover all frequent itemsets with 2 items and above, we have applied the Apriori algorithm described in [1]. This algorithm consists of several passes over the transactions. Pass *k* finds all frequent *k*-itemsets (itemsets of size k). It uses the set of frequent (k-1) itemsets found in the previous pass to narrow the search for k-itemsets. Figure 7 gives a pseudo code for the Apriori algorithm [1] where minimum support is 1%.

```
Apriori Algorithm

Input: Var Smin% = 1%
        $L_{k-1}$ = The set all frequent 1-itemsets
Output: The set of frequent itemsets

1: For (k=2; $L_{k-1}$!= null; k++) {
2:    Generate $C_k$, the set of candidate k-itemsets, from $L_{k-1}$, the set of frequent (k-
      1) Itemsets found in the previous steps;
3:
4:    Scan the transactions to count the occurrences of itemsets in $C_k$;
5:
6:    Find $L_k$, a subset of $C_k$ containing k-itemsets with counts no less than 1:
7:  (Smin% * total number of transactions);
8:}
9:
10:Return $L_1$ U $L_2$ U $L_3$ U .... $L_k$
11:
12: To generate candidate. From $L_{k-1}$ to $C_k$
13:    insert into Ck
14:    select p.item1, p.item2, ..., p,item(k-1), q.item(k-1)
15:    from $L_{k-1}$ p, $L_{k-1}$ q
16:    where p.item$_1$ = q.item $_2$ AND ...
17:            p.item $_{k-2}$ = q.item $_{k-2}$
18:            p.item $_{k-1}$ < q.item $_{k-2}$
19:
20:    For each itemset c in $C_k$;
21:     For each item X in c:
22:             If c – {X} is not in $L_{k-1}$ then:
23:                     Delete c from $C_k$
```

Figure 7. Apriori Algorithm [1]

Line 1 – 10 is the pseudo-code for Apriori. On line 2, it generates the candidate k using the method on line 12-23. Line 12-18 forms candidate k-itemsets by joining frequent (k-1) itemsets. Line 20-23 ensures all itemsets in the candidate k-itemsets is frequent. On line 18, it ensures that the candidate itemsets generated has no duplicates.

4.3.2 Path Traversal Mining Technique Implementation

To build the graphs to mine for path traversal pattern, we need to find all the maximal forward references (MFR). To find the MFR from the user transactions, we used the technique described in [4]. Briefly, maximal forward reference is a user access path without backward reference. Backward reference means revisiting a previously visited page by the same user access. Backward reference is mainly provided for ease of traveling but not for browsing, and in web usage mining, we want to concentrate on the discovery of forward patterns. The maximal forward reference algorithm described in [4] will filter out the effect of backward references. The following is an example of finding maximal forward reference from a user transaction:

Transaction of a user: {A, B, C, D, C, B, E, G, H, G, W, A, O, U, O, V}
MFRs found: {ABCD, ABEGH, ABEGW, AOU, AOV}

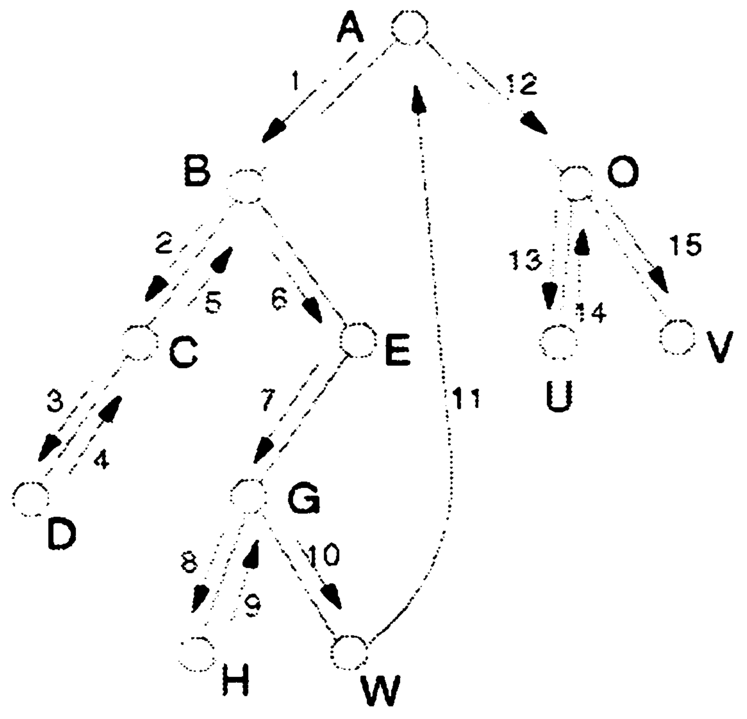


Figure 8. An Example of Discovering Maximal Forward Reference [4]

In Figure 8, some nodes in the graph might be revisited because of its location, rather than its content. When a backward reference occurs, a forward reference path terminates. The resulting forward reference is called a MFR. The following is an algorithm that finds MFRs [4]:

Maximal Forward Reference

```
Input: The set user transactions
Output: The set of MFRs written into the database

For each user transaction:

Step 1: Set i= 1, string Y = null for initialization, and flag F = 1

Step 2: Let A =  si and B = di.
        If A is equal to null then
        /* this is the beginning of a new traversal */
        begin
           If (Y != null) then
                  write Y to the database Df;
                  Set string Y = B;
                  Go to Step 4.
           End
        End

Step 3: If B is equal to some reference (say the j-th reference) in string Y
        then
           /* this is a cross-referencing back to a previous reference */
           begin
                  If F is equal to 1 then write out string Y to database;
                         Discard all the references after the j-th one in string Y;
                  F = 0;
                  Go to Step 4.
           End
        else, append B to the end of string Y.
           /* we are continuing a forward traversal */
           If F is equal to 0, set F = 1.
        End
Step 4: Set i = i + 1. If the sequence is not completed scanned then go to Step 2.
```

Figure 9. Maximal Forward Reference Algorithm [4]

A user transaction is a traversal sequence of the form {(s1, d1), (s2, d2), ... (sn, dn)} where s is the source and d is the destination. On line 3, *i* is the index of the current path in the transaction and *Y* is used to store the current forward reference path. Also, the flag F is set to 1 to indicate a forward traversal. On line 5, *si* and *di* indicate the current traversal path and the associated sets A, as the source, and B, as the destination.

Compute Graph

With the resulting MFRs, we can now compute the graph (a tree) that represents the physical layout of a web site, with web pages as nodes and hypertext links between pages as directed edges. On each node of the tree, there is a count to indicate the number of users who accessed the node. Figure 10 shows the algorithm to build the tree. After the tree is generated, nodes that do not pass the minimum support are pruned away.

With this graph, we can identify which paths are frequently visited with respect to a confidence value.

```
Build tree from MFRs

Input: The set of MFRs
Output: A tree

Var MFR = The set of MFRs
Var parent = 0
Var minSupportCount = Smin% * number of MFR
Step 1: Create a tree with a dummy node call root at level 0

Step 2: For i = 0; i < MFR.size (); i++
            MaximalForwardReference A = MFR [i];
              parent = root
            For j = 0; j < A's number of pages; j++
                  Page page = A.getPageOnPosition(j)
                   If the tree level j contains a node with this page's name
                  Increase the count of this node
                        parent = this node
                   Else
                        Create a node with this page's name
                        Add this new node under the parent
                        parent = this new node
                        End If
             End For
        End For

Step 3: For each node in the tree
              If node's count < minSupportCount then
                   Remove node and its child.
              End If
         End For
```

Figure 10. Tree building algorithms using Maximal Forward References.

For example, if MFR = {(A, B, C), (A, D, E)}, then the graph will look like the following:

```
ROOT  ---A (2) --- B (1) --- C (1)
          |
          ---------D(1)------E (1)
```

Naturally, not all "sections" in a web site will be relevant or interesting to every user of WUM; therefore we can use the "section" interested by the user as a constraint. The idea behind this constraint is that user normally has something in mind and his/her search is focused, which is modeled through constraint awareness WUM. This constraint driven mining prunes away all irrelevant information. Obviously, we achieve efficiency with a smaller set of information to perform mining.

With the MFR algorithm, we are able to mine for general usage pattern of a web site, but we will not be able to mine for usage pattern of a specific part in a web site. The is because when the minimum support is based on the number of occurrences in all the paths traversed for all parts of the web site, it may prune away paths traversed for that specific part. We can still mine for user access patterns for a particular part using MFR algorithm if we set the minimum support low enough, so that it will not prune away paths traversed for that particular part. However, if the minimum support is set too low, we will end up having too much information to analyse at the pattern analysis phase. Therefore, to mine for usage pattern of a specific part, we have added *context awareness* constraint into the MFR algorithm.

First, we will only consider user transactions that contain at least one page having the constraint in it. The constraint being one or more values in the page name of the access log records. For example, a constraint can be "comp353". Using this, we can make sure that all the user transactions considered are meaningful and relevant for the usage pattern of that specific constraint. Also, the concept of when a forward reference terminates is redefined. The forward reference will terminate if a backward reference occurs, or, when a page not containing the constraint occurs. This latter refinement in the definition will better capture the destination page for that particular section. These two constraints have been added based on the fact that web pages that are closely related usually contain some similar pattern in their page name. For instance, all pages related to comp353 will contain comp353 somewhere in their URLs. Figure 11 shows the pseudo code of the MFR algorithm with constraint.

Maximal Forward Reference with context awareness constraint

Input: The set user transactions and the constraint
Output: A set of MFRs written into the database

```
For each user transaction with at least one page containing the constraint:

Var isContainSatisfy = false
Var constraint = X

Step 1: Set i = 1, string Y = null for initialization, and flag F = 1

Step 2: Let A = si and B = di.
        If A is equal to null then
          /* this is the beginning of a new traversal */
          begin
            If (Y != null) then
                write Y to the database Df;
                Set string Y = B;
               If B containConstraint X then
                  IsContainSatisfy = true
               End
                Go to Step 4.
            End
        End

Step 3: If B is equal to some reference (say the j-th reference) in string Y
           or (isContainSatisfy and B not containConstraint X)
            then
            /* this is a cross-referencing back to a previous reference */
            begin
                    If F is equal to 1 then write out string Y to database Df;
                            Discard all the references after the jth one in string Y;
                    F = 0;
                    Go to Step 5.
            End
         Else, append B to the end of string Y.
               /* we are continuing a forward traversal */
               If B containConstraint X, set isContainSatisfy = true
               If F is equal to 0, set F = 1.
        End

Step 4: Set i = i + 1. If the sequence is not completed scanned then go to Step 2.
```

Figure 11. Maximal Forward Reference algorithm with context awareness constraint

An example of the application of this algorithm is to mine for the usage pattern of the comp353 course in the CS web site. The constraint here will be "comp353". First, only user transactions having at least one page containing "comp353" will be considered. Then, in the MFR algorithm, in addition to filtering backward references we will also filter pages that do not contain the "comp353" because the user probably went to a page that is not part of the comp353 course. Here is a possible scenario:

User A accesses pages A, B, comp353/C.html, D, E, comp353/F.html.

In this case, it will generate 2 maximal forward references:
1) A, B, comp353/C.html
2) D, E, comp353/F.html

5 Experimentation

We used the user access logs from a CS (Computer Science) department's Web Server to carry out our experiment. We used WUM to mine for usage knowledge for this web site.

5.1 Pre-processing of access logs

The user access logs for the CS web site that were made available to this project contains about 2,500,000 entries from February 20, 2002 to March 8, 2002. Initially, these entries are stored in a log file. We restored them in a database for processing in a later phase. Each access log contains the following information: IP address, time, method URL protocol, status, size, and agent. However, since we do not need all these information for web usage mining, we extracted the required information, which includes: the IP address, time, and page name. The page name is extracted from the method URL protocol.

As explained in section 2.2, we need to clean the data before storing them into the database. We filtered the access logs against some filtering criteria and stored the rest into the database. The filtering criteria are pages that contains either one of the following extensions gif, jpeg, GIF, JPEG, jpg, JPG, or have status code that is not in the two hundreds levels range. As a result of data cleaning phase, 480,266 access logs considered clean and were inserted into the AccessLog table in the database. Then, we identified user transactions by grouping access logs with the same IP and that are created within 30 minutes. The number of user transactions identified is 57,807 records. Figure 12 shows a summary of the pre-processing phase.

Description of Pre-processing Results	**Number of records**
Number of raw user access logs from Feb.20, 2002 to March 8, 2002	2,570,515 logs
Number of user access logs after removing non-user initiated access	500,388 logs
Number of user access logs after removing unsuccessful requests	480,266 logs
Number of user transactions (30 min session time) identified from the cleaned user access logs	57,807 transactions

Figure 12. Summary of the Pre-processing Phase

5.2 Usage pattern mining

After the user transactions are identified, we applied the two data mining techniques: an association rule mining technique and a path mining technique to discover the usage patterns.

5.2.1 Association Rule Mining

First, we applied the association rule mining technique with some minimum support s%. The value s we considered were 50, 30, 20, 10, 5, 1 and 0.8. The accessed are distributed among a large number of items, and some were accessed about 1000 times, which is about 1% of the transactions. Therefore, we picked 1% as the minimum support for finding association rules in this case study. Figures 13, 14, and 15 show the frequent itemsets found as the result. Note, "/" means www.cs.concordia.ca and there is no itemset for 4 items because there is no candidates generated for 4 items.

item1	count
/	44040
/current_students.shtml	6720
/programs/ugrad/courses.html	1444
/people/people.html	1411
/~teaching/comp691l/	1390
/~faculty/grogono/tunick.html	1323
/~teaching/comp239/2002W/	1263
/people/faculty.html	1208
/~teaching/comp218/	1142
/~teaching/comp691l/html/	1140

Figure 13. Frequent Itemset with 1 item

item1	Item2	count
/	/current_students.shtml	1577
/people/faculty.html	/people/people.html	852
/	/people/people.html	464
/	/programs/ugrad/courses.html	407
/	/people/faculty.html	355
/~teaching/comp691l/	/~teaching/comp691l/html/	327
/~teaching/comp239/2002W/	/current_students.shtml	304
/current_students.shtml	/people/people.html	249
/~teaching/comp691l/	/current_students.shtml	221
/	/~teaching/comp691l/	164
/	/~teaching/comp239/2002W/	150

Figure 14. Frequent Itemset with 2 items

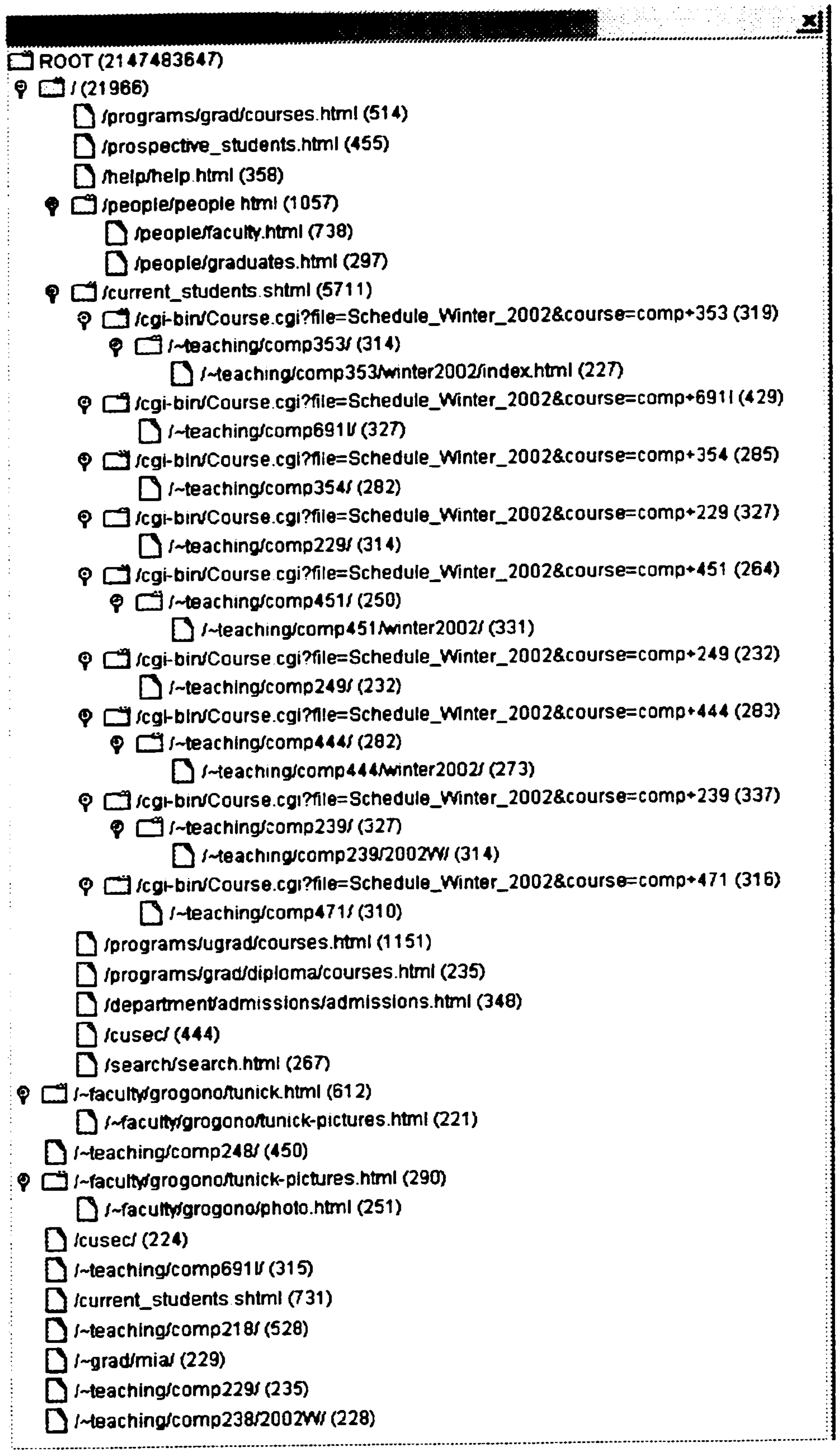
item1	item2	Item3	count
/	/people/faculty.html	/people/people.html	305
/	/current_students.shtml	/people/people.html	126
/~teaching/comp691l/	/~teaching/comp691l/html/	/current_students.shtml	117

Figure 15. Frequent Itemset with 3 items

A frequent itemsets is a set of items that happen frequently together. The itemset with 1 item is basically all the most accessed items. The itemset with 2 or more items are items that are accessed together frequently inside a transaction. We will use these frequent itemsets to generate association rules in section 5.1.

5.2.1 Path Traversal Mining

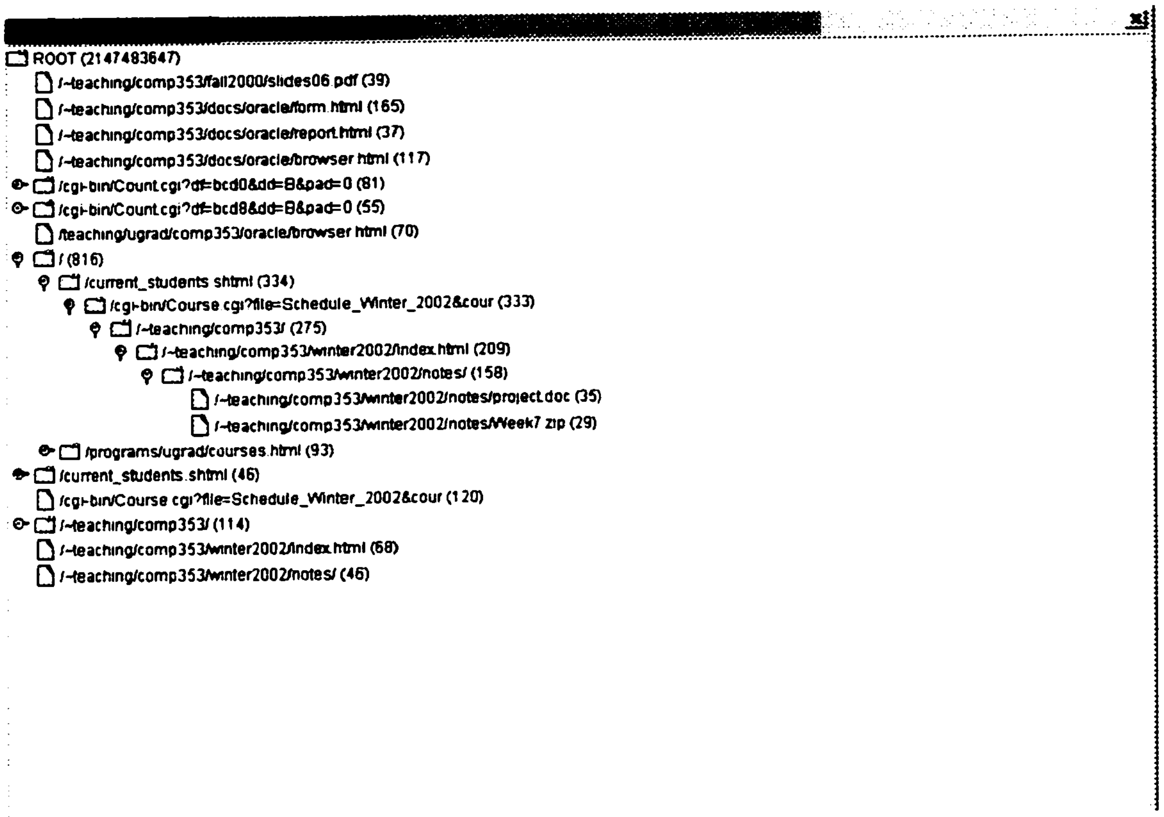
We applied the general path mining technique with a minimum support of 0.5% to obtain a set of maximal forward references. The various s values we considered were 50, 30, 20, 10, 5, 1 and 0.5. The frequently accessed pages were accessed about 300 times, which is about 0.5% of the transactions. Therefore, we picked 0.5% as the minimum support for finding frequently accessed paths. Figure 18 illustrates the tree generated from this set of maximal forward references.



```
ROOT (2147483647)
  / (21966)
      /programs/grad/courses.html (514)
      /prospective_students.html (455)
      /help/help.html (358)
    /people/people.html (1057)
        /people/faculty.html (738)
        /people/graduates.html (297)
    /current_students.shtml (5711)
      /cgi-bin/Course.cgi?file=Schedule_Winter_2002&course=comp+353 (319)
        /~teaching/comp353/ (314)
            /~teaching/comp353/winter2002/index.html (227)
      /cgi-bin/Course.cgi?file=Schedule_Winter_2002&course=comp+6911 (429)
          /~teaching/comp6911/ (327)
      /cgi-bin/Course.cgi?file=Schedule_Winter_2002&course=comp+354 (285)
          /~teaching/comp354/ (282)
      /cgi-bin/Course.cgi?file=Schedule_Winter_2002&course=comp+229 (327)
          /~teaching/comp229/ (314)
      /cgi-bin/Course.cgi?file=Schedule_Winter_2002&course=comp+451 (264)
        /~teaching/comp451/ (250)
            /~teaching/comp451/winter2002/ (331)
      /cgi-bin/Course.cgi?file=Schedule_Winter_2002&course=comp+249 (232)
          /~teaching/comp249/ (232)
      /cgi-bin/Course.cgi?file=Schedule_Winter_2002&course=comp+444 (283)
        /~teaching/comp444/ (282)
            /~teaching/comp444/winter2002/ (273)
      /cgi-bin/Course.cgi?file=Schedule_Winter_2002&course=comp+239 (337)
        /~teaching/comp239/ (327)
            /~teaching/comp239/2002W/ (314)
      /cgi-bin/Course.cgi?file=Schedule_Winter_2002&course=comp+471 (316)
          /~teaching/comp471/ (310)
      /programs/ugrad/courses.html (1151)
      /programs/grad/diploma/courses.html (235)
      /department/admissions/admissions.html (348)
      /cusec/ (444)
      /search/search.html (267)
  /~faculty/grogono/tunick.html (612)
      /~faculty/grogono/tunick-pictures.html (221)
    /~teaching/comp248/ (450)
  /~faculty/grogono/tunick-pictures.html (290)
      /~faculty/grogono/photo.html (251)
    /cusec/ (224)
    /~teaching/comp6911/ (315)
    /current_students.shtml (731)
    /~teaching/comp218/ (528)
    /~grad/mia/ (229)
    /~teaching/comp229/ (235)
    /~teaching/comp238/2002W/ (228)
```

Figure 16. Resulting Tree for the CS web site with a support of 0.5%

We then applied the path mining technique with a contextual constraint of "comp353" and the same minimum support, of 0.5%. Figure 10 is the tree generated from the set of maximal forward references found after applying this technique.



```
ROOT (2147483647)
    /~teaching/comp353/fall2000/slides06.pdf (39)
    /~teaching/comp353/docs/oracle/form.html (165)
    /~teaching/comp353/docs/oracle/report.html (37)
    /~teaching/comp353/docs/oracle/browser.html (117)
    /cgi-bin/Count.cgi?df=bcd0&dd=B&pad=0 (81)
    /cgi-bin/Count.cgi?df=bcd8&dd=B&pad=0 (55)
    /teaching/ugrad/comp353/oracle/browser.html (70)
    / (816)
        /current_students.shtml (334)
            /cgi-bin/Course.cgi?file=Schedule_Winter_2002&cour (333)
                /~teaching/comp353/ (275)
                    /~teaching/comp353/winter2002/index.html (209)
                        /~teaching/comp353/winter2002/notes/ (158)
                            /~teaching/comp353/winter2002/notes/project.doc (35)
                            /~teaching/comp353/winter2002/notes/Week7.zip (29)
        /programs/ugrad/courses.html (93)
    /current_students.shtml (46)
    /cgi-bin/Course.cgi?file=Schedule_Winter_2002&cour (120)
    /~teaching/comp353/ (114)
    /~teaching/comp353/winter2002/index.html (68)
    /~teaching/comp353/winter2002/notes/ (46)
```

Figure 17. Resulting tree for Comp353 with min support of 0.5%

Figure 16 and 17 show two trees with web pages as nodes and hypertext links between pages as directed edges. For these two trees, a dummy node call ROOT is created and is set as the root node. On each node, there is a count to show the number of times a page was accessed in the path.

6 Results and their Analysis

With the results obtained from the two data mining techniques applied, we can obtain useful rules, patterns, and statistics that can be used to provide recommendations to the owner of the Computer Science's department in our case study.

6.1 Knowledge from Association Rules Results

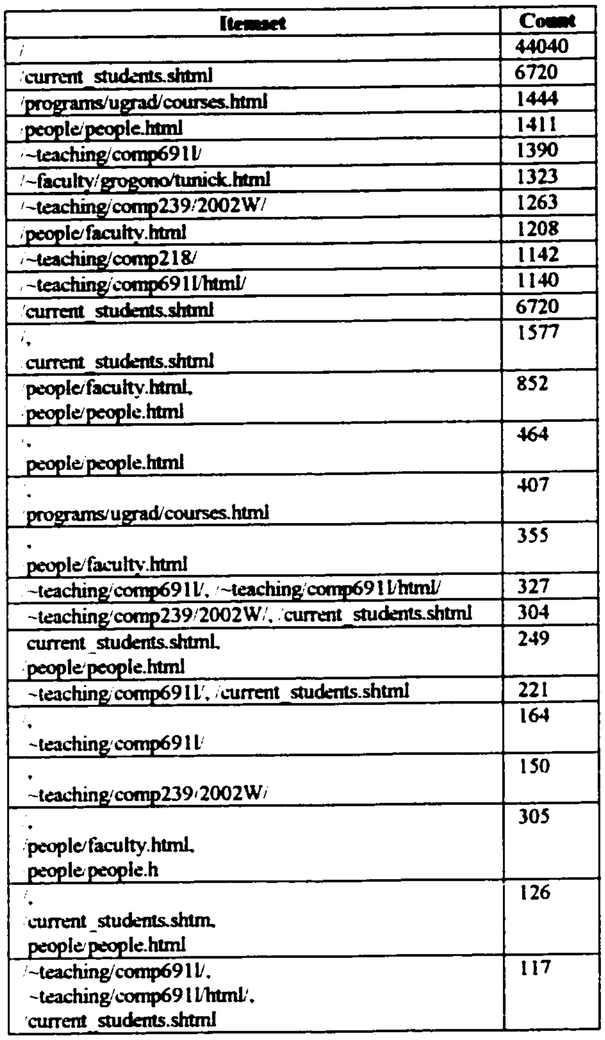
With the frequent itemsets found, we can determine association rules for the Computer Science web. Here is the algorithm from [1] that we used to generate these rules:

Association rule algorithm

Input: Frequent itemsets and the confidence value Cmin%
Output: A set of association rules.

```
1: For each itemsets i:
2:       For each nonempty proper subset s of i:
3:             If confidence = count(i) / count(s) >= Cmin % then:
4:                   Output s => ( i – s);
```

Figure 18. Association rule algorithm



Itemset	Count
/	44040
/current_students.shtml	6720
/programs/ugrad/courses.html	1444
/people/people.html	1411
/~teaching/comp691I/	1390
/~faculty/grogono/tunick.html	1323
/~teaching/comp239/2002W/	1263
/people/faculty.html	1208
/~teaching/comp218/	1142
/~teaching/comp691I/html/	1140
/current_students.shtml	6720
/, /current_students.shtml	1577
/people/faculty.html, /people/people.html	852
/, /people/people.html	464
/, /programs/ugrad/courses.html	407
/, /people/faculty.html	355
/~teaching/comp691I/, /~teaching/comp691I/html/	327
/~teaching/comp239/2002W/, /current_students.shtml	304
/current_students.shtml, /people/people.html	249
/~teaching/comp691I/, /current_students.shtml	221
/, /~teaching/comp691I/	164
/, /~teaching/comp239/2002W/	150
/, /people/faculty.html, /people/people.h	305
/, /current_students.shtm, /people/people.html	126
/~teaching/comp691I/, /~teaching/comp691I/html/, /current_students.shtml	117

Figure 19. Frequent Itemsets

Examples of association rules for i = { /, /current_students.shtm, /people/people.html }

1. / , /current_students.shtm => /people/people.html (confidence 126/1577 = 8%)
2. /, /people/people.htm => /current_students.shtm (confidence 126/464= 27 %)
3. /current_students.shtm, /people/people.html => / (confidence 126/249= 50 %)
4. / => /current_students.shtm, /people/people.html (confidence 126/44040= 0.3%)
5. /current_students.shtm => /, /people/people.html (confidence 126/6720= 1.8%)
6. /people/people.html => /, /current_students.shtm (confidence 126/1411= 9%)

Also, with the frequent itemsets, we can determine which items were frequently accessed. For instance, if the owner of the site wants to know which 5 pages were most frequently accessed, then by looking at the frequent itemsets, we can conclude that /, /current_students.shtml, /programs/ugrad/courses.html, /people/people.html, and /~teaching/comp691l/ are the top 5 items accessed. Other association rules could also be discovered from the frequent itemsets, but since it is not the goal of this study to list all the rules that could be generated from frequent itemsets, therefore, we will not discuss them.

6.2 Knowledge from Path Traversal Results

With the graphs generated from the path traversal pattern results, we can determine the frequently accessed paths. We consider a path frequent if it is visited with a confidence of at least 1% in 57807 transactions that were found from the pre-processing phase. Here are the frequently accessed paths:

1. page / was visited 21966 times (confidence 21966 / 57807 = 38%)
2. page /, and /people/people.html were visited 1057 times (confidence 1057 / 57807 = 1.8%)
3. page /, and /current_students.shtml were visited 5711 times (confidence 5711 / 57807 = 9.8%)

For the specific graphs generated for a course, for example comp353, we have determined the frequent accessed paths. We consider a path frequent if it is visited with a confidence of at least 1% in 2182 transactions related to the constraint comp353. Here are the frequently accessed paths:

1. page / was visited 816 times (confidence 816 / 2182 = 37%)

2. page /, and /current_students.shtml were visited 334 (confidence 334 / 2182 = 15%)

3. page /, and /current_students.shtml, /cgi-bin/Course.cgi?=file=Schedule_Winter_2002&cour were visited 333 (confidence 333/ 2182 = 15%)

4. page /, and /current_students.shtml, /cgi-bin/Course.cgi?=file=Schedule_Winter_2002&cour, /~teaching/comp353/ were visited 275 (confidence 275 / 2182 = 12.7%)

5. page /, /current_students.shtml, /cgi-bin/Course.cgi?=file=Schedule_Winter_2002&cour, /~teaching/comp353/, and /~teaching/comp353/winter2002/index.html were visited 209 (confidence 209 / 2182 = ~1%).

With the graphs generated for a course, for example comp353, we also discovered the following rules:

1. 70 % of the users who accessed /~teaching/comp353/ do so by starting at page /. This was concluded with the following facts:
    a. /~teaching/comp353/ were accessed 389 times
    b. 275 times of those accessed started from page /
    c. 114 times of those accessed started from other page

2. 57 % of the users who accessed /~teaching/comp353winter2002/index.html accessed /~teaching/comp353winter2002/notes/ right after. This was concluded with the following facts:
    a. /~teaching/comp353winter2002/index.html were accessed 277 times
    b. /~teaching/comp353winter2002/notes/ were accessed 158 times right after /~teaching/comp353winter2002/index.html was accessed.

6.3 Recommendations

From the few rules generated as described in sections 6.1 and 6.2, there are already some recommendations that can be made for the designer of the CS department's web site.

First, with the association rules generated, we can dynamically provide links for what is likely to be accessed next in a highly visible area, so that it will help a user navigate the web site. For example, if a user accessed /current_students.shtml and /people/people.html, then we can dynamically add / into the highly visible area since we are 50% confident that the user may access / later in the transaction. An algorithm to generate dynamic hypertext links from user access patterns can be found in [5].

Second, with the knowledge of mostly accessed pages, we can concentrate putting important messages on those pages. For instance, we can put important message on the top five accessed page which are /, /current_students.shtml, /programs/ugrad/courses.html, /people/people.html, and /~teaching/comp691l/.

Finally, with the knowledge of the most traversed paths, we can provide some hot links to get to the destination of such paths or put important messages along these paths. For example, we can provide hot links in pages /~teaching/comp353winter2002/index.html through /~teaching/comp353winter2002/notes/ since we know that 57% of users who accessed index.html also accessed "notes" right after.

6.4 Time measures for WUM

The following is the time measures for the various WUM phases:

Description of process	Time Taken (sec)
Pre-processing 1 - Convert Raw Log File to Access Log (includes filtering process)	21600
Pre-processing 2 - User transactions identification	1020
Pattern Discovery 1 - Association rule (Candidate and Itemset)	380
Pattern Discovery 2 - Maximum Forward Reference (including Tree Generation)	1341
Pattern Discovery 3 - Maximum Forward Reference with constraint (including Tree Generation)	180
Pattern Analysis – Finding Rules	110

Figure 20. Time measures of various WUM phases

From Figure 20, we can see that the most time consuming phase in web usage mining is the pre-processing phase. Therefore, it is very important that we define the "correct" filters, so that we can prune away as much as possible the unwanted information in the access logs. Also, the time taken to do MFR with constraint is just 13% of time of the general MFR. If the interest of the MFR is to find usage knowledge of a particular part of a web site, then it is definitely more efficient to use MFR with constraint as opposed to MFR

7 Conclusions

In this work, we studied the web usage mining process. We designed and implemented a system (WUM) to do web usage mining. We presented our results of the experimentation of doing web usage mining with WUM for a CS department's web site. Using the knowledge obtained, we are able to recommend changes to the structure of the CS department's web site to improve its usability. In particular, we can make suggestions to put important messages and announcements at frequently accessed pages, or provide dynamic links that shows what pages are most accessed next.

To mine for the usage knowledge, we applied two data mining techniques: Association rule, Path Mining. To solve the problem of mining for a specific part in a large web site, we introduced an extension to the maximal forward reference algorithm that has *context awareness constraint*. With this extension, we were able to mine for usage pattern for a part of a web site more efficiently. From Figure 20, we can see that we saved 83% of the time when we use this extension. Also, the concept of when a forward reference terminates were redefined to better suit the analyses of the usage of a specific part of a web site. With this extension, we were able to filter out irrelevant user transactions in the access log to a particular part of the web site.

As a future work, we would like to provide in the WUM system a library of mining techniques for sequential patterns and allow users to choose their favourite technique(s) to apply on a desired set of user transactions. This will provide a high-level generic environment to study, apply, and compare various mining techniques. In addition, the extended system can be used to study the effect of an optimization technique more readily and conveniently

8 References

[1] Agarwal. R. and Srikant R., "Fast algorithms for mining association rules", *In Proceedings of the 20th VLDB conference*, pp.487-499, Santiago, Chile, 1994.

[2] Agarwal. R.and Srikant R., "Mining Sequential Patterms", *Proc. 11th Int'l Conf. Data Eng.*, pp.3-14, mar.1995.

[3] Cooley R., Mobasher. B., and Srivastava J., "Data preparation for mining World Wide Web browsing patterns", *Journal of Knowledge and Information Systems*, (1) 1, 1999.

[4] Chen, M. S., J. S. Park, et al., "Data Mining for Path Traversal Patterns in a Web Environment", *In Proceedings of 16th International Conference on Distributed Computing Systems*, 1996.

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

[6] B. Mobasher, R. Cooley, and J. Srivastava., "Automatic personalization based on Web usage mining", *In Communications of the ACM*, (43) 8, August 2000.

[7] Cooley, R., B. Mobasher, et al., "Grouping Web Page References into Transactions for Mining World Wide Web Browsing Patterns", *In Proceedings of Knowledge and Data Engineering Workshop, Newport Beach,* CA, IEEE, 1997.

[8] Cooley, R., B. Mobasher, et al., "Data Preparation for Mining World Wide Web Browsing Patterns", *Knowledge and Information Systems*, 1(1), 1999.

[9] P. Pirolli, J. Pitkow, and R. Rao., "Silk from a sow's ear: Extracting usable structures from the web". *In Proc. of 1996 Conference on Human Factors in Computing Systems (CHI-96)*, Vancouver, British Columbia, Canada, 1996.

[10] M. Mehta, R. Agrawal, and J. Rissanen., "SLIQ: A fast scalable classifier for data mining.", *In Proc. of the Fifth Int'l Conference on Extending Database Technology*, Avignon, France, 1996.

[11] K. Wu, P. S. Yu, and A. Ballman., "Speedtracer: A web usage mining and analysis tool.", *IBM Systems Journal*, 37(1):89-105, 1998.