Web Mining: Pattern Discovery from World Wide Web Transactions *

Bamshad Mobasher, Namit Jain, Eui-Hong (Sam) Han, Jaideep Srivastava
{mobasher,njain,han,srivasta}@cs.umn.edu
Department of Computer Science
University of Minnesota
4-192 EECS Bldg., 200 Union St. SE
Minneapolis, MN 55455, USA

March 8, 1997

Abstract

Web-based organizations often generate and collect large volumes of data in their daily operations. Analyzing such data can help these organizations to determine the life time value of clients, design cross marketing strategies across products and services, evaluate the effectiveness of promotional campaigns, and find the most effective logical structure for their Web space. This type of analysis involves the discovery of meaningful relationships from a large collection of primarily unstructured data, often stored in Web server access logs. We propose a framework for *Web mining*, the applications of data mining and knowledge discovery techniques to data collected in World Wide Web transactions. We present data and transaction models for various Web mining tasks such as the discovery of association rules and sequential patterns from the Web data. We also present a Web mining system, WEBMINER, which has been implemented based upon the proposed framework, and discuss our experimental results on real-world Web data using the WEBMINER.

Keywords: data mining, world wide web, association rules, sequential patterns, web mining.

*This work was supported in part by NSF grant ASC-9634719, Army Research Office contract DA/DAAH04-95-1-0538, Cray Research Inc. Fellowship, and IBM partnership award, the content of which does not necessarily reflect the policy of the government, and no official endorsement should be inferred. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute, Cray Research Inc., and NSF grant CDA-9414015. See http://www.cs.umn.edu/~han/papers.html#DataMiningPapers for other related papers.

1 Introduction and Background

As more organizations rely on the World Wide Web to conduct business, traditional strategies and techniques for market analysis need to be revisited. Organizations collect large volumes of data and analyze it to determine the life time value of customers, cross marketing strategies across products, and effectiveness of promotional campaigns. In the Web, such information is generally gathered automatically by Web servers and collected in server or access logs. Analysis of server access data can provide information on how to restructure a Web site for increased effectiveness, better management of workgroup communication, and analyzing user access patterns to target ads to specific groups of users. Most existing Web analysis tools [Inc96, eSI95, net96] provide very primitive mechanisms for reporting user activity, i.e. it is possible to determine the number of accesses to individual files, the times of visits, and URLs of users. However, these tools usually provide little analysis of data relationships among the accessed files, which is essential to fully utilizing the data gathered in daily transactions. A comprehensive analysis tool must automatically discover such relationships among users accesses.

In this paper we describe a framework for the application of two data mining techniques, i.e. discovery of association rules and sequential patterns to extract relationships from data collected by Web servers. Recently, several researchers have proposed the application of data mining techniques to facilitate information discovery on global information systems such as the Internet [ZH95, KKS96]. The focus of these proposals is knowledge discovery across the Internet, based on content, and not the analysis of user access patterns on various Web servers. Web server access logs, however, have been used as a testbed for the application of certain data mining tasks such as the discovery of frequent episodes [MTV95]. Recently, *maximal forward references* have been proposed [CPY96] as a way to extract meaningful user access sequences. *Web mining* is the application of data mining techniques to large Web data repositories, examples of which are provided below.

Discovering Association Rules: In of Web mining, an example of an association rule is the correlation among accesses to various files on a server by a given client. For example, using association rule discovery techniques we can find the following correlations: (i) 60% of clients who accessed the page with URL `/company/products/`, also accessed the page `/company/products/product1.html`; (ii) 40% of clients who accessed the Web page with URL `/company/products/product1.html`, also accessed `/company/products/product2.html`; and (iii) 30% of clients who accessed `/company/special-offer.html`, placed an online order in `/company/products/product1`. In Web mining additional properties of data can be used to prune the search space, since information about a site's structural hierarchy can be used. For example, if the support for `/company/products/` is low, one may conclude that the search for association between the two secondary pages with URLs `/company/products/product1` and `/company1/products/product2` should be pruned since neither are likely to have adequate support.

Discovery of Sequential Patterns: Given a database of time-stamped transactions, the problem of discovering sequential patterns [MTV95, SA96] is to find inter-transaction patterns, i.e. the presence of a set of items followed by another item, in the time-stamp ordered transaction set. In Web server transaction logs, a visit by a client is recorded over a period of time. By analyzing this information, we can determine temporal relationships among data items such as: (i) 30% of clients who visited `/company/products/product1.html`, had done a search in Yahoo, within the past week on keywords w_1 and w_2; and (ii) 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. Another important kind of information we may be interested in is the common characteristics of all clients that visited a particular file within the time period $[t_1, t_2]$. Alternatively, we may be interested in a time interval, i.e. an hour, day or week, during which a particular file is most accessed.

As the examples above show, mining for knowledge from web log data has the potential of revealing information of great value. While this certainly is an application of existing data mining algorithms, e.g. discovery of association rules or temporal sequences, the overall task is not one of simply adapting existing algorithms to new data. Because of many unique characteristics of the client-server model in the World Wide Web, including radical differences between the physical and logical data organizations of web repositories, it is necessary to develop a new framework to enable the mining process. Specifically, there are a number of issues in *pre-processing data for mining* that must be addressed before the mining algorithms can be run. These include developing a model of web log data, developing techniques to clean/filter the raw data to eliminate outliers and/or irrelevant items, grouping individual page accesses into semantic units (i.e. transactions), and specializing generic data mining algorithms to take advantage of the specific nature of web log data.

In this paper we present an architecture for a web mining system called the WEBMINER, and our initial experiences with it. Our specific contributions include (i) development of a flexible architecture for web mining, (ii) developing a model for a *user transaction* which consists of multiple log entries, (iii) clustering algorithms for grouping log entries into transactions, (iv) adaptation of association rule and temporal sequence discovery algorithms to web mining, and (v) experimental evaluation of the system.

The rest of this paper is organized as follows: Section 2 discusses the architecture of the Web mining process. Section 3 presents data and transaction models for Web mining. Section 4 we present the implementation of the WEBMINER, and also present our experimental results with the WEBMINER. Finally, in Section 5, we look at future and continuing research and development issues.

2 An Architecture for Web Mining

A Web server access log contains a complete history of file accesses by clients. Most WWW access logs follow the *Common Log Format* specified as part of the HTTP protocol by CERN and NCSA [Luo95]. A log entry, following this standard, contains the client IP address, user id, access time, request method, and the URL of the page accessed, the protocol used for data transmission, an error code, and the number of bytes transmitted. Table 1 shows a snapshot of a portion of the WWW access log for the main Web server in the Computer Science Department at the University of Minnesota.

The primary objective of Web mining is to discover interesting patterns in accesses to various Web pages within the Web space associated with a particular server. In order to successfully apply generic data mining techniques to web data, one must first transform this data to a suitable form. In particular, unlike market basket analysis, where a single transaction is defined naturally according to customer purchase activity, in Web data the notion of a transaction must be defined based on the properties of the application domain.

Our proposed architecture, thus divides the web mining process into two main parts. The first part includes the domain dependent processes of transforming the Web data into suitable "transaction" form, and the second part includes the, largely domain independent, application of generic data mining techniques (such as the discovery of association rule and sequential patterns) as part of the system's data mining engine. The overall architecture for the Web mining process is depicted in Figure 1.

Generally, there are a variety of files accessed as a result of a request by a client to view a

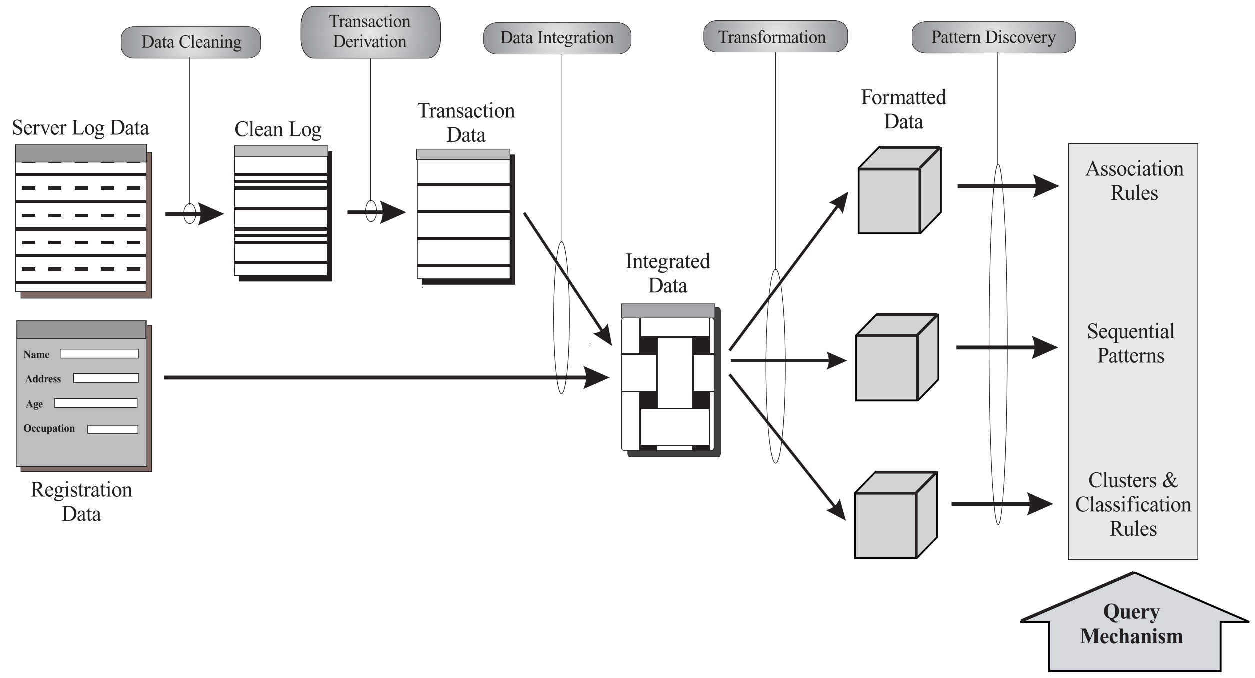

Figure 1: General Architecture for WEBMINER

```
...
looney.cs.umn.edu han - [09/Aug/1996:09:53:52 -0500] "GET /~mobasher/courses/cs5106/cs5106l1.html HTTP/1.0" 200 9370
mega.cs.umn.edu njain - [09/Aug/1996:09:53:52 -0500] "GET / HTTP/1.0" 200 3291
mega.cs.umn.edu njain - [09/Aug/1996:09:53:53 -0500] "GET /images/backgnds/paper.gif HTTP/1.0" 200 3014
mega.cs.umn.edu njain - [09/Aug/1996:09:53:53 -0500] "GET /images/misc/footer.jpg HTTP/1.0" 200 13355
mega.cs.umn.edu njain - [09/Aug/1996:09:54:12 -0500] "GET /cgi-bin/Count.cgi?df=CS-home.dat&dd=C&ft=1 HTTP/1.0" 200 646
mega.cs.umn.edu njain - [09/Aug/1996:09:54:18 -0500] "GET /~advisor HTTP/1.0" 302
mega.cs.umn.edu njain - [09/Aug/1996:09:54:19 -0500] "GET /~advisor/ HTTP/1.0" 200 487
looney.cs.umn.edu han - [09/Aug/1996:09:54:28 -0500] "GET /~mobasher/courses/cs5106/cs5106l2.html HTTP/1.0" 200 14072
mega.cs.umn.edu njain - [09/Aug/1996:09:54:31 -0500] "GET /~advisor/csci-faq.html HTTP/1.0" 200 13786
looney.cs.umn.edu han - [09/Aug/1996:09:54:47 -0500] "GET /~mobasher/courses/cs5106/princip.html HTTP/1.0" 200 6965
moose.cs.umn.edu mobasher - [09/Aug/1996:09:55:50 -0500] "GET /~suharyon/lisa.html HTTP/1.0" 200 654
moose.cs.umn.edu mobasher - [09/Aug/1996:09:55:53 -0500] "GET /~suharyon/line/line16.gif HTTP/1.0" 200 1423
moose.cs.umn.edu mobasher - [09/Aug/1996:09:55:57 -0500] "GET /~suharyon/joko1.jpg HTTP/1.0" 200 30890
...
```

Table 1: Sample Entries from a Web Server Access Log

particular Web page. These include image, sound, and video files; executable cgi files; coordinates of clickable regions in image map files; and HTML files. Thus, the server logs contain many entries that are redundant or irrelevant for the data mining tasks. For example, all the image file entries are irrelevant or redundant, since as a URL with several image files is selected, the images are transferred to the client machine and these files are recorded in the log file as independent entries. We call the process of removing redundant or irrelevant entries from the Web server log files *data cleaning*. Data cleaning is performed by checking the suffix of the URL name. For instance, all the log entries with filename suffixes such as, gif, jpeg, GIF, JPEG, jpg, JPG and map are removed from the log.

After the data cleaning, the log entries must be partitioned into logical clusters that represent a single user transaction. One of the significant factors which distinguish Web mining from other data mining activities is the method used for identifying user transactions. This process, which we call *transaction derivation* is particularly important in the context of web data, because it allows the discovery phase to focus on relevant access points of a particular user rather than the intermediate pages accessed for navigational reasons.

Recently, *maximal forward references* have been proposed [CPY96] as a way to extract meaningful user access sequences. However, this method does not consider that some of the "backward" references may provide useful information or that not all forward references are meaningful for the discovery process. We prefer to cluster several user references (entries in the clean log file) into what would be considered a single user transaction. These transaction clusters are formed either statically based on a user specified time gap, or dynamically, according to time distance between entries, time spent on a particular page relative to its size, logical structure of the Web space, or other possibly application dependent criteria. The transaction model for WEBMINER is presented in the Section 3.

As depicted in Figure 1, access log data may not be the only source of data for the Web mining process. User registration data, for example, is playing an increasingly important role, particularly as more security and privacy conscious client-side applications restrict server access to a variety of information, such as the client IP addresses or user IDs. The data collected through user registration must then be integrated with the access log data. While WEBMINER currently does not incorporate user registration data, we are exploring various data integration issues in the context of Web mining. For a study of data integration in databases see [LHS$^+$95].

Once the domain-dependent data transformation phase is completed, the resulting transaction data must be formatted to conform to the data model of the appropriate data mining task. For instance, the format of the data for the association rule discovery task may be different than the format necessary for mining sequential patterns. The Webmining data model for association rules and sequential patterns are described in the next section. This allows for the application of generic data mining algorithms to the Web transaction data.

Finally, a query mechanism will allow the user (analyst) to provide more control over the discovery process by specifying various constraints. The emerging data mining tools and systems lead naturally to the demand for a powerful data mining query language, on top of which many interactive and flexible graphical user interfaces can be developed [HFW$^+$96]. Most of the conventional query languages are constrained by a schema, but in our case, the data model does not fall in this category. Recently, several query languages have been proposed which are not constrained by a schema. DMQL [HFW$^+$96], UnQL [BDS95, BDHS96] and Lorel [QRY$^+$95] fall into this category, and can be easily extended to query the World Wide Web. Some guidelines for a good data mining language were proposed in [HFW$^+$96], which among other things, highlighted the need for specifying the exact data set and various thresholds in a query. Such a query mechanism can provide user control over the data mining process and allow the user to extract only relevant and useful rules. In WEBMINER, we have implemented a simple Query mechanism by adding some primitives to an SQL-like language. This allows the user to provide guidance to the mining engine by specifying the patterns of interest.

As an example, consider a situation where the user is interested in the patterns which start with URL A, and contain B and C in that order, this pattern can be expressed as a regular expression $A * B * C *$. To see how this expression is used within a SQL-like query, suppose further that the analyst is interested in finding all such rules with a minimum support of 1 % and a minimum confidence of 90 %. Moreover, let us assume that the analyst is interested only in clients from the domain `.edu`, and only wants to consider data later than Jan 1, 1996. The query based on these parameters can be expressed as follows:

```
SELECT association-rules(A*B*C*)
FROM   log.data
WHERE  date ≥ 960101 AND domain = edu AND
       support = 1.0 AND confidence = 90.0
```

This information from the query is used to reduce the scope, and thus the cost of the mining process. The development of a more general query mechanism along with appropriate Web-based user interfaces and visualization techniques are issues relevant to the future development of the WEBMINER.

3 A Transaction Model for Web Mining

Unlike market basket analysis, where a *market basket transaction* is defined as the set of items bought by a customer in a single purchase, there is no natural definition of a user transaction in a web browsing scenario. For example, at one extreme we could consider each log entry to be a separate transaction, while at the other we could consider all the entries in the log made by a particular user as a single transaction. Since a transaction is used to model a *useful unit of work*, either of these choices is probably too extreme. We believe that any approach to group web log entries into transactions must use information about both the *nature of data* and the *type of analysis* to be done on it. We propose to use such information in a 2-step process. In the first step we use clustering as a general approach to grouping web log entries into transactions. The clustering is based on comparing pairs of log entries and determining the *similarity* between them by means of some kind of distance measure(s). Entries that are sufficiently close are grouped together. In the second step, we use information about the type of analysis and specialize the groups formed in step 1 into transactions suited to the specific analysis. In the following, we first describe our general clustering approach and its present implementation. Next, we show how information about the type of analysis, *viz.* association rule discovery and temporal sequence discovery, can be used to develop the respective transactions.

3.1 Clustering Log Entries into Groups

Let L be a set of server access log entries. A log entry $l \in L$ includes the client IP address $l.ip$, the client user id $l.uid$, the URL of the accessed page $l.url$, and the time of access $l.time$. There are other fields in web log entries, such as the request method used (e.g., POST or GET) and the size of the file transmitted. For our present illustration however, we only focus on the fields listed above.

In using clustering to determine the similarity of two log entries, i.e. whether they belong to the same group, distance metrics on many different attributes can be defined. Determining an appropriate set of attributes to cluster on, and defining appropriate distance metrics for them is an important problem, and is being addressed in our ongoing research. As an initial cut, we have used only the time dimension for clustering of log entries. This is because time is the principal dimension that captures the nature of web browsing, since this process is essentially sequential in nature. Further, as illustrated in definition 1, the log entries are laid out on the temporal dimension, and a time window is used for the grouping, where successive entries in a group can be at most a maximum time gap Δt apart. In addition, we consider partition the log entries based on IP address and user ID, since we are interested in browsing patterns of individual users.

Definition 1 A *Log Entry Group (LEG)* g is a triple:

$$g =< ip_g, uid_g, \{(l_1^g.url, l_1^g.time), \ldots, (l_m^g.url, l_m^g.time)\} >$$

where, for $1 \leq k \leq m$, $l_k^g \in L$, $l_k^g.ip = ip_g$, $l_k^g.uid = uid_g$, and

$$l_{k+1}^g.time - l_k^g.time \leq \Delta t$$

As an example consider the log entries of Table 1. If the user-specified maximum time gap Δ is 1 minute, then the transaction belonging to the client `njain` (starting from time `09/Aug/1996:09:53:52`), will be the URL set $\{/, /\sim\texttt{adviser}, /\sim\texttt{adviser}/\texttt{csci}-\texttt{faq.html}\}$.

3.2 Extracting Association Transactions From LEGs

Determining association rules does not need any temporal information. Hence, in extracting *association transactions* from the log entry groups, we filter out the time information.

Definition 2 An *association transaction* t is a triple extracted from a log entry group g, so that $t =< ip_g, uid_g, \{l^g_1.url, \ldots, l^g_m.url, \} >$ (note that $l^g_{k+1}.time - l^g_k.time \leq \Delta t$).

Let T be the set of all association transactions of the form $< ip_t, uid_t, URL_t >$ (as defined above), where $URL_t = \{l^t_1.url, \ldots, l^t_m.url\}$. We define the *web space*, WS, associated with the access log as $WS = \bigcup_{t \in T} URL_t$. We define the *support count* for a set of URLs $U \subseteq WS$ to be $\sigma(U) = |\{t | U \subseteq URL_t\}|$. In other words, the support count for U is the number of times (within the access log) that the URLs in U have been accessed by clients in one transaction. We can now formally define the notion of an association rule in the context of Web mining.

An *association rule* is an expression of the form $X \overset{s,\alpha}{\Longrightarrow} Y$, where $X \subseteq WS$ and $Y \subseteq WS$. The *support* s of the rule $X \overset{s,\alpha}{\Longrightarrow} Y$ is defined as $\sigma(X \cup Y)/|T|$, and the confidence α is defined as $\sigma(X \cup Y)/\sigma(X)$. The task of discovering an association rule is to find all rules $X \overset{s,\alpha}{\Longrightarrow} Y$, where s is at least a given threshold and α is at least another given threshold. For example, the rule

$$\{\texttt{/company/products/}, \texttt{/company/products/product1.html}\} \overset{0.01,0.75}{\Longrightarrow} \texttt{/company/products/product2.html}$$

indicates that 75% of clients who accessed the "products" section of the Web site and connected to the page for "product1" in that section, also visited the page for "product2", and that this combination of events occurred in 1% of all transactions.

3.3 Extracting Temporal Transactions From LEGs

For the task of mining sequential patterns, we need to keep track of the access time for each of the URLs accessed within that transaction. Furthermore, we are interested in the activity of a particular client spanning the whole access log. We define a temporal transaction to be the set of all the URL names and their access times for the same client where successive log entries are within a user-specified time gap Δt. We also associate a unique time stamp with each such transaction. In other words, a transaction is simply an LEG with a computed time stamp.

Definition 3 A *temporal transaction* t is a 4-tuple $t =< ip_t, uid_t, UT_t, time(t) >$ where

$$UT_t =< (l^t_1.url, l^t_1.time), \ldots, (l^t_m.url, l^t_m.time) >,$$

such that for $1 \leq k \leq m$, $l^t_k \in L$, $l^t_k.ip = ip_t$, $l^t_k.uid = uid_t$, $l^t_{k+1}.time - l^t_k.time \leq \Delta t$, and $time(t) = \max_{1 \leq i \leq m} l^t_i.time$.

Let T be the set of all temporal transactions. For each transaction $t =< ip_t, uid_t, UT_t > \in T$, we call UT_t the *URL–Time Set* (*UT–Set*) for t. A *UT–Sequence* is a list of UT–Sets ordered according to transaction times. In other words, given a set $T' = \{t_i \in T | 1 \leq i \leq k\}$ of transactions, a UT–Sequence S for T' is: $S =< UT_{t_1}, \ldots, UT_{t_k} >$, where $time(t_i) < time(t_{i+1})$ for $1 \leq i \leq k-1$.

Given a client c with ip address ip and user id u, let T_c be the set of all temporal transactions involving that client. So, $T_c = \{t \in T | ip_t = ip \text{ and } uid_t = u\}$. The UT–Sequence for T_c is a special sequence, S_c, called the *client–sequence* for c, composed of all the UT–Sets of transactions involving a client c. In other words,

$$S_c = < UT_{t_1}, UT_{t_2}, \ldots, UT_{t_n} >$$

where for $1 \leq i \leq n$, $t_i \in T_c$.

Definition 4 A UT–Sequence $A = < a_1, a_2, \ldots, a_n >$ is a *subsequence* of another UT–Sequence $B = < b_1, b_2, \ldots, b_m >$, denoted $A \sqsubseteq B$, if there exists integers $i_1 < i_2 < \ldots < i_n$, such that $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \ldots, a_n \subseteq b_{i_n}$.

Let ID be the set of all client ids (ip and uid pair) in the Web server access log. Then the *support count* for a UT–Sequence S, denoted $\sigma(S)$, is:

$$\sigma(S) = |\{S_c | c \in ID \text{ and } S \sqsubseteq S_c\}|.$$

Intuitively, the support count for the UT–sequence S is the number of client sequences that support S (i.e., include S as a subsequence). Now, given UT–sequences $S = X \cdot Y$ (where $\cdot$ denotes concatenation of two sequences), a *sequential pattern* can be defined as an expression of the form $X \overset{s,\alpha}{\Longrightarrow} Y$, where the *support* s of the rule $X \overset{s,\alpha}{\Longrightarrow} Y$ is defined as $\sigma(X \cdot Y)/|ID|$, and the confidence α is defined as $\sigma(X \cdot Y)/\sigma(X)$.

The task of Web mining for sequential patterns is to find all rules $X \overset{s,\alpha}{\Longrightarrow} Y$ (among all UT–sequences $S = X \cdot Y$), where s is at least a given threshold and α is at least another given threshold.

4 Experimental Evaluation

In this section we describe the implementation of WEBMINER, a Web mining system based on the framework presented in the previous sections and our experience with it. We give some of the algorithms we have used in the implementation of techniques for discovering association rules and sequential patterns from Web data.

We have run these algorithms on the access log of Cray Research Home Page located at http://www.cray.com for the experiment. The log contains about 520K entries corresponding to the requests made during May of 1996 and its size is about 56M Bytes.

4.1 Mining Association Rules

Given the access log of Cray Research Home Page, we used max time gap of 10 minutes to have 44K transactions of size 670K Bytes. There were 3686 distinct URLs referenced in the transaction. Given the transactions, the problem of mining association rules is to generate all association rules that have support and confidence greater than the user-specified minimum support (called *minsup*) and minimum confidence (called *minconf*) respectively. We have used *apriori* algorithm presented in [AS94] for mining association rules. We experimented with support between 0.05% and 1.0% and confidence of 80%. The experiment was performed on a SUN SPARC Station 5 with 70 MHz CPU, 32M Bytes main memory, 510M Bytes disk drive and SunOS Release 5.4. Table 2 shows some examples of the rules discovered.

The first two are singleton rules without an antecedent. Rule 1 shows that 1.23% of transactions contain the Liquid-cooled Cray T3E[1] System home page, while rule 2 shows that 0.68% of transactions contain the Air-cooled Cray T3E Systems. This result is different from the results

[1]Trade Mark of Cray Research Inc.

Rule No.	Confidence(%)	Support(%)	Association Rules
1	100.00	1.23	`/PUBLIC/product-info/T3E/LC_T3E.html`
2	100.00	0.68	`/PUBLIC/product-info/T3E/AC_T3E.html`
3	82.83	3.17	`/PUBLIC/product-info/T3E` $\Longrightarrow$ `/PUBLIC/product-info/T3E/CRAY_T3E.html`
4	90.00	0.14	`/PUBLIC/product-info/J90/J90.html` `/PUBLIC/product-info/T3E` $\Longrightarrow$ `/PUBLIC/product-info/T3E/CRAY_T3E.html`
5	97.18	0.15	`/` `/PUBLIC/product-info/J90` `/PUBLIC/product-info/T3E/CRAY_T3E.html` `/PUBLIC/product-info/T90` $\Longrightarrow$ `/PUBLIC/product-info/T3E` `/PUBLIC/sc.html`

Table 2: Some Examples of the Association Rules Discovered

from other tools that provide statistical summary on the access log. Other tools do not have the concept of transactions as defined here. As a result, if one client accesses one site several times in a short span of time, simple statistics collection tools would report the hit several times. On the other hand, in our system, the hit would be recorded only once because all of the accesses belong to one transaction and the association transaction is defined as a set of items.

Rule 3 says that 82.83% of clients that accessed the URL `/PUBLIC/product-info/T3E`, also visited `/PUBLIC/product-info/T3E/CRAY_T3E.html` which is under `/PUBLIC/product-info/T3E`. Rule 4, on the other hand, shows that 90% of clients that accessed `/PUBLIC/product-info/T3E` and `/PUBLIC/product-info/J90/J90.html`, also visited `/PUBLIC/product-info/T3E/CRAY_T3E.html`. These two rules demonstrate that clients who access J90[2] home page and T3E top page visit T3E main home page (`CRAY_T3E.html`) about 7% more than other clients who just accessed the T3E main home page.

Based on the discussion of Section 1, we can already observe how these discovered rules can be useful. For example, the combinations of rules 3 and 4 in Table 2 might suggest that there is a portion of the content of the J90 page that encourages clients to go back and access the T3E page. By, moving or copying this portion to a higher level in the hierarchy (e.g., `/PUBLIC/product-info/T3`) we might be able to increase the overall support for rule 2 in the table.

4.2 Mining Sequential Patterns

In the implementation of WEBMINER, the time gap defining transactions for sequential patterns, defined in section 3, is taken to be zero. Thus, UT-sequences are made up of singleton sets of items and each item is a URL accessed by a client in a transaction.

We have modified the *apriori* algorithm to mine frequent sequential patterns. Note that the support for a pattern now depends on the ordering of the items, which was not true for association rules. For example, a transaction consisting of URLs ABCD in that order contains BC as an subsequence, but does not contain CB. So, all permutations have to be generated. This is even more computationally intensive than generating all combinations which has already been reported

[2]Trade Mark of Cray Research Inc.

Pattern No.	Support(%)	Sequential Patterns
1	5.63	`/PUBLIC/sc.html` → `/PUBLIC/product-info/T3E/AC_T3E.html`
2	2.69	`/PUBLIC/sc.html` → `/PUBLIC/product-info/T3E/AC_T3E.html` → `/PUBLIC/product-info/T3E/quotes.html`
3	2.89	`/PUBLIC/sc.html` → `/PUBLIC/product-info/T90/BAYER.html` → `/PUBLIC/product-info/T90/T90apps.html`
4	0.93	`/PUBLIC/product-info/RM/convergence9.html` → `/PUBLIC/product-info/RM/images` → `/PUBLIC/product-info/RM/hw7.html` → `/PUBLIC/product-info/RM/unifying6.html`

Table 3: Some Examples of the Sequential Patterns Discovered

to be difficult [AS94]. For getting around this problem, we have changed the joining operation for candidate generation itself. In generating candidates of size $k+1$, instead of matching the first $k-1$ elements of two frequent itemsets, we match the last $k-1$ elements of the first frequent sequential pattern of size k with the first $k-1$ elements of the second frequent sequential pattern of size k.

For finding the support of candidate sequential patterns, the question is: Given a UT-sequence $< A_1, \dots, A_n >$, how can we determine if there exists a subsequence $< B_1, \dots, B_k >$ within a certain time span TS? The important factor to consider is that the number of occurrences of that subsequence within the time span is insignificant. The only thing that matters is whether the subsequence is present or not. Figure 2 gives the desired algorithm. The basic idea behind the algorithm is to keep the latest starting times of all the prefixes of the subsequence. The algorithm takes a transaction along with its time stamp, a subsequence, and a time span (TS) as input.

```
1. Initialize prefixtime to -1;
2. for (all items i in transaction) do begin
3.    if (firstitem of subsequence matches) then
4.       prefixtime[1] = transtime[i];
5.    for ( all other items j in subsequence ) do
6.       if (item j of subsequence matches) then
7.          prefixtime[j] = prefixtime [j - 1];
8.    if (prefixtime[last] > 0 && transtime[i] - prefixtime[last] ≤ TS) then
9.       return TRUE // subsequence found.
10. end
11. return FALSE // subsequence not found.
```

Figure 2: Algorithm gen_count

We have performed an experiment to find frequent sequential patterns from the access log of Cray Research Home Page using the above algorithm. For this experiment, we did not consider the confidence of the sequential patterns. Table 3 shows some examples of the sequential patterns discovered.

Pattern 1 shows that 5.63% of the clients accessed the Supercomputing Systems home page (`sc.html`) followed by Air-cooled Cray T3E systems home page. Pattern 2 shows that 2.69% of the clients went on to check out the customer quotes on the T3E system after accessing the two URLs of

Pattern 1. Pattern 3 demonstrates the sequential pattern of clients accessing the Supercomputing Systems home page, the page containing a story on T90[3] technical solution for Bayer group and the page containing application vendor quotes, in that order.

Discovered sequential patterns can be used to predict user actions or provide suggestions for restructuring a site. For example, if one of the patterns in Table 3 is not desirable, then direct links can be placed in corresponding pages in order to redirect client traffic patterns. Or, if a pattern has a high confidence, it may be useful to put certain kinds of information (e.g., advertisements) in the predicted path of clients corresponding to that pattern.

5 Conclusions and Future Directions

In this paper we have presented a framework for Web mining, the application of data mining and knowledge discovery techniques to WWW server access logs. This framework includes a flexible architecture distinguishing the domain specific data transformation tasks from the generic data mining engine. We have also described WEBMINER, a system based on the proposed framework, and presented experimental results on real-world industrial data to illustrate some of its applications (Section 4).

Currently, we are extending the implementation of WEBMINER to incorporate mechanisms for clustering analysis and discovery of classification rules. The query mechanism in WEBMINER will also be extended to include clustering and classification constraints. Also an important area of ongoing research is to find better methods of clustering log entries into user transactions, including using criteria such as time differential among entries, time spent on a page relative to the page size, and user profile information collected during user registration.

Another interesting area of future work involves the development of autonomous agents that analyze the discovered rules to provide meaningful courses of action or suggestions to users (for instance to make suggestions about modifying the organization of content within the Web space, to automatically prefetch Web files based on discovered rules and user profiles, or to present various users with dynamically generated content based on user patterns).

Other areas of future work include developing a more flexible query mechanism which can handle both pre-discovery queries (on the data) and post-discovery queries (on the rules); integrating user registration data with the access log data in the discovery process; and extending the WEBMINER to process Web data distributed across several servers each collecting their own access log and user registration data.

References

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference*, pages 487–499, Santiago, Chile, 1994.

[BDHS96] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proc. of 1996 ACM-SIGMOD Int. Conf. on Management of Data*, 1996.

[BDS95] P. Buneman, S. Davidson, and D. Suciu. Programming constrcuts for unstructured data. In *Proceedings of ICDT'95, Gubbio, Italy*, 1995.

[3] Trade Mark of Cray Research Inc.

[CPY96] M.S. Chen, J.S. Park, and P.S. Yu. Data mining for path traversal patterns in a web environment. In *Proceedings of the 16th International Conference on Distrib uted Computing Systems*, pages 385–392, 1996.

[eSI95] e.g. Software Inc. Webtrends. *http://www.webtrends.com*, 1995.

[HFW$^+$96] J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaiane. Dmql: A data mining query language for relational databases. In *SIGMOD'96 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'96)*, Montreal, Canada, 1996.

[Inc96] Open Market Inc. Open market web reporter. *http://www.openmarket.com*, 1996.

[KKS96] I. Khosla, B. Kuhn, and N. Soparkar. Database search using informatiuon mining. In *Proc. of 1996 ACM-SIGMOD Int. Conf. on Management of Data*, Montreal, Quebec, 1996.

[LHS$^+$95] E. Lim, S.Y. Hwang, J. Srivastava, D. Clements, and M. Ganesh. Myriad: Design and implementaion of federated database prototype. *Software – Practive & Experience*, 25(5):533–562, 1995.

[Luo95] A. Luotonen. The common log file format. *http://www.w3.org/pub/WWW/*, 1995.

[MTV95] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 210–215, Montreal, Quebec, 1995.

[net96] net.Genesis. net.analysis desktop. *http://www.netgen.com*, 1996.

[QRY$^+$95] D. Quass, A. Rajaraman, Y.Sagiv, J. Ullman, and J. Widom. Querying semistructured heterogeneous information. In *International Conference on Deductive and Object Oriented Databases*, 1995.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[SA96] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. of the Fifth Int'l Conference on Extending Database Technology*, Avignon, France, 1996.

[ZH95] O. R. Zaane and J. Han. Resource and knowledge discovery in global information systems: A preliminary design and experiment. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 331–336, Montreal, Quebec, 1995.