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
Minneapolis, MN 55455, USA

Technical Report 96-050 (September 3, 1996)

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, knowledge discovery, world wide web, association rules, sequential patterns, web mining.

1 Introduction and Background

As more organizations rely on the Internet and the World Wide Web to conduct business, the traditional strategies and techniques for market analysis need to be revisited in this context. 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 customers, cross marketing strategies across products, and effectiveness of promotional campaigns, among other things. However, such analyses involve the discovery of meaningful relationships from a large collection of, primarily, unstructured data.

Most of this information is usually generated in the daily activity of organizations as client or customer transaction logs. An example is the daily transaction logs generated in supermarkets containing bar code data. In the context of the World Wide Web such information is generally gathered automatically by Web servers and collected in server or access logs. For organizations engaged in commerce on the World Wide Web, mining these logs for valuable information is just as essential as it is in more traditional data collection mechanisms. Analysis of server access data can also provide valuable information on how to better structure a Web site in order to create a more effective presence for the organization. In organizations using intranet technologies, such analysis can shed important information on more effective management of workgroup communication and organizational infrastructure. Finally, for organizations that sell advertising on the World Wide Web, analyzing user access patterns helps in targeting ads to specific groups of users.

Most of the existing Web analysis tools [Inc96, eSI95, net96] provide mechanisms for reporting user activity in the servers and various forms of data filtering. Using such tools, for example, it is possible to determine the number of accesses to the server and the individual files within the organization's Web space, the times or time intervals of visits, and domain names and the URLs of users of the Web server. However, in general, these tools are designed to deal with low to moderate traffic servers, and furthermore, they usually provide little or no analysis of data relationships among the accessed files and directories within the Web space.

Analysis and discovery of various data relationships is, of course, essential in fully utilizing the valuable data gathered in daily transactions. Thus, a comprehensive analysis tool must be able to automatically discover such data relationships including the correlations among Web pages, sequential patterns over time intervals, and classification of users according to their access patterns. In addition, such a tool must be able to discover relationships in very high traffic servers with very large (possibly distributed) access logs.

In this paper we describe a framework for the applications of various data mining and knowledge discovery techniques to data collected by Web servers. Data mining [FPSM91, MCPS93, FPSS96] is the process of extracting valid and interesting relationships from large collections of data using AI or statistical techniques. There are a variety of data mining techniques that have been proposed, including, clustering [KR90, Fis95, NH94], classification [HKS$^+$96, MAR96, CS96, HCC93, WK91], discovery of association rules [AS94, HS95, SON95, SA95], and discovery of sequential patterns [MTV95, SA96].

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 based on content across the Internet 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].

Web mining is the application of these and other data mining techniques to large Web data repositories. This, however, is not simply an engineering task of 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, including data models, architectures, and algorithms, to facilitate the successful implementation of Web mining tools. Here, we present such a framework. We also present our Web mining system, WEBMINER, that implements several data mining techniques in this context and discuss our experimental results.

The rest of this paper is organized as follows. Section 2 provides an overview of Web mining and its applications. Section 3 discusses the structure of data and various data cleaning and preprocessing tasks necessary to prepare Web data for mining. We also present data and transaction models for various Web mining tasks. In Section 4 we provide details of the algorithms for the different mining techniques implemented in the WEBMINER, and also discuss a simple query mechanism that can provide user control over the Web mining process. In Section 5 we present our experimental results with the WEBMINER, and finally, in Section 6, we look at future and continuing research and development issues.

2 Web Mining and Its Applications

In this section we discuss some of the most important data mining techniques and their applications in the context of discovering hidden patterns and relationships within the Web data. We can place these data mining techniques in 3 general categories: discovery of association rules, discovery of sequential patterns, and discovery of classification rules and data clusters. In the following, we briefly discuss each of these techniques and what they mean in the context of Web mining.

2.1 Discovering Association Rules

The association rule discovery techniques [AS94, HS95, SON95, SA95] are generally applied to databases of transactions where each transaction consists of a set of items. In such a framework the problem is to discover all associations and correlations among data items where the presence of one set of items in a transaction implies (with a certain degree of confidence) the presence of other items.

In the context of Web mining, this problem amounts to discovering the correlations among accesses to various files available on the server by a given client. Each transaction is comprised of a set of URLs accessed by a client in one visit to the server. For example, using association rule discovery techniques we can find correlations such as the following:

- 60% of clients who accessed the page with URL `/company/products/`, also accessed the page `/company/products/product1.html`; or
- 40% of clients who accessed the Web page with URL `/company/products/product1.html`, also accessed `/company/products/product2.html`; or
- 30% of clients who accessed `/company/announcements/special-offer.html`, placed an on-line order in `/company/products/product1`.

Since usually such transaction-based databases contain extremely large amounts of data, current association rule discovery techniques try to prune the search space according to *support* for items under consideration. Support is a measure based on the number of occurrences of the items within transaction logs. In Web mining there are additional properties of the data that can be used to further prune the search space. This is partly due to the fact that, generally, the files associated with a particular site are organized in a hierarchical fashion, and the structure of this hierarchy is known before hand.

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 such rules for organizations engaged in electronic commerce can help in the development of effective marketing strategies. But, in addition, association rules discovered from WWW access logs can give an indication of how to best organize the organization's Web space. For example, if one discovers 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 that this information should be moved to a higher level (e.g., `/company/products`) to increase access to `file2.html`.

2.2 Time Sequences and Discovery of Sequential Patterns

Generally, transaction-based databases collect data over a period of time, and the time-stamp for each transaction is explicitly available. Given such a database of transactions, the problem of discovering sequential patterns [MTV95, SA96] is to find inter-transaction patterns such that the presence of a set of items is 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. The time stamp associated with a transaction in this case will be a time interval which is determined and attached to the transaction during the data cleansing process.

By analyzing this information, the WEBMINER can determine temporal relationships among data items such as the following:

- 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; or
- 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 data dependency that can be discovered, using the temporal characteristics of the data, are similar time sequences. For example, we may be interested in finding common characteristics of all clients that visited a particular file within the time period $[t_1, t_2]$. Or, conversely, we may be interested in a time interval (within a day, or within a week, etc.) in which a particular file is most accessed.

Generally, the techniques used in sequential pattern discovery are similar to those used in association rule discovery or classification, except in this case the discovered rules have to be further classified based on the time stamp information. The discovery of sequential patterns in Web server access logs allows Web-based organizations to predict user visit patterns and helps in targeting advertising aimed at groups of users based on these patterns.

2.3 Classification and Clustering

Another common data mining technique involves the discovery of hidden common patterns among data items and their classification according to these patterns [MAR96, CS96, HCC93, WK91]. Classification allows one to develop a profile for items belonging to a particular group according to their common attributes. This profile can then be used to classify new data items that are added to the database. Currently most classification algorithms utilize inductive techniques such as tree induction or neural induction.

In Web mining, classification techniques allow one to develop a profile for clients who access particular server files based on demographic information available on those clients. The client-server model of the World Wide Web allows for the collection of valuable demographic data on clients without even the use of surveys or other instruments usually used to obtain such information. Much of this information on clients can be obtained by analyzing client requests and the information transmitted by the client browser, including the URL.

Classification on WWW access logs allows one to discover relationships such as the following:

- clients who often access `/company/products/product3` tend to be from educational institutions; or
- clients from state or government agencies who visit the site tend to be interested in the page `/company/products/product1.html`; or
- clients who placed an online order in `/company/products/product2`, tend to have previously visited the site for Company X; or
- 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.

In some cases, valuable information about the clients can be gathered by the server automatically from the client browsers. This includes information available on the client side in the history files, cookie files, etc. Other methods used to obtain profile and demographic information on clients include user registration, online survey forms, and techniques such as "anonymous ticketing" [Inc96].

Furthermore, obtaining profile information on clients and discovering classifications of data items (server files), allows one to group together clients or data items that have similar characteristics. In data mining this kind of activity is referred to as *clustering* [KR90, Fis95, NH94]. Clustering of client information or data items on Web transaction logs, can facilitate the development and execution of future marketing strategies, both online and off-line, such as automated return mail to clients falling within a certain cluster, or dynamically changing a particular site for a client, on a return visit, based on past classification of that client.

3 A Data Model for Web Mining

3.1 Structure of Access Log Data

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.

Generally, there are a variety of files accessed as a result of a request by a client to view a particular Web page. These include a variety of image, sound, and video files; executable cgi files; coordinates of clickable regions in image map files; and HTML files. Typical examples of files accessed as a result of a client request include:

- `<filename>.html`: the HTML file for the requested page;
- `<filename>.gif` or `<filename>.jpg`: image files.
- `<filename>.map?<x,y>`: file mapped to coordinates x and y of an image map file;
- `<program>.cgi?<arguments>`: a server-side executable file.

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. The server logs contain entries that are redundant or irrelevant for these data mining tasks. For example, all the image file entries are irrelevant or redundant. 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. For our purposes, image files without a hyper link can be ignored because they are not used to visit other pages. An image file with a hyper link is redundant since as a client selects the hyper link, the destination URL will be recorded in the log. A similar situation exists with regards to image map files and other multimedia support files.

Unless the task is to determine which display is better to attract the clients to visit certain URLs, we can remove all these redundant log entries for the Web mining tasks. We call this process *data cleansing*. Data cleansing is performed by checking the suffix of the URL name. All the URL

```
...
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

entries with filename suffixes such as, gif, jpeg, GIF, JPEG, jpg, JPG and map are removed from the log. While we do not consider the requests for executable files here, in general such requests may not result in redundant or irrelevant entries in the log, as many cgi programs generate HTML files on the fly, which are then transmitted to the client.

Data cleansing process also involves the selection of a subset of the fields that are relevant for the data mining tasks. The fields of interest are IP address, user id, access time and the URL.

After the log entries are cleaned, the log data is converted into a form suitable for a specific data mining task. This transformation is accomplished according to the transaction model for that particular task (e.g., association rule or sequential pattern discovery). These data and transaction models are described in the following sections.

Figure 1 shows the overall architecture of the WEBMINER. As shown in Figure 1, before any knowledge discovery takes place from the Web data, the data goes through a preprocessing phase to clean the data from irrelevant or redundant entries. Then the data is formatted appropriately according to the application (Association Rules and Sequential Patterns require the input data to be in different forms). In order to provide a greater degree of user control, a query language has been built on top of the basic engine. The user can specify the type of pattern to look for, and only those are retrieved. Details of the query language are given in Section 4.

3.2 Data and Transaction Model for Association Rules

Unlike market basket analysis, where a single transaction is defined naturally, we don't have a natural definition of transaction for the task of mining association rules. We define a transaction based on the set of all log entries belonging to the same client (IP address and user id), within a

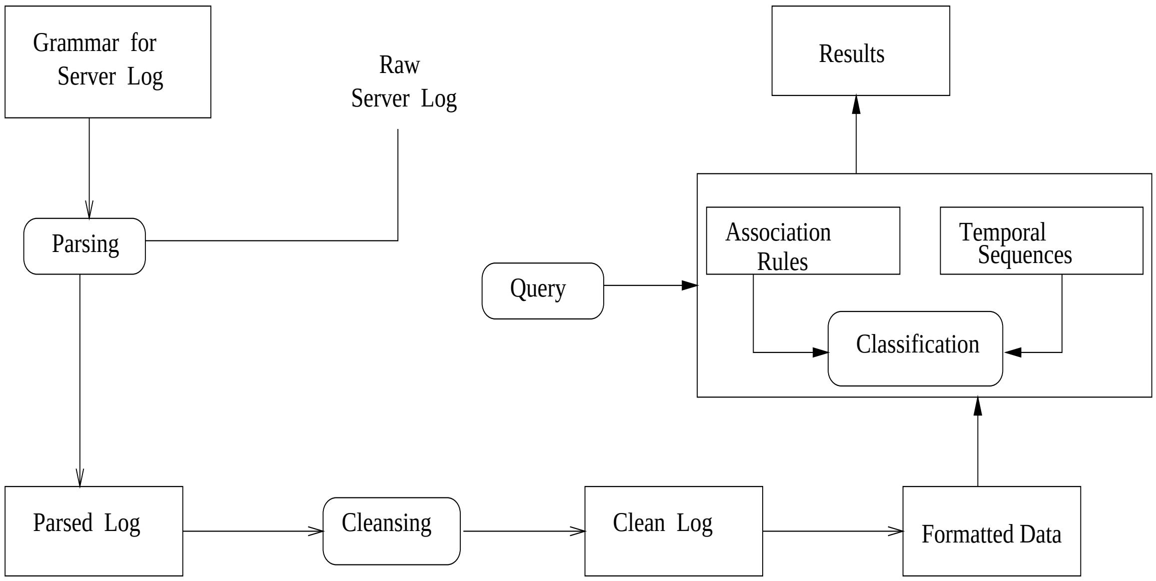

Figure 1: Architecture of WEBMINER

given maximum time gap (provided by the user).

Let L be a set of server access log entries. A log entry $l \in L$ has the following components:

- the IP address of the client, denoted $l.ip$;
- the user id for the client, denoted $l.uid$;
- the URL of the page accessed by the client, denoted $l.url$; and
- the time of access, $l.time$.

Note that there are other fields in WWW log entries based on the Common Log Format, such as the request method used (e.g., POST or GET) and the size of the file transmitted. For our purposes, we only focus on the above essential components in each entry.

Definition 1 An *association transaction* t is a triple:

$$t = < ip_t, uid_t, \{l_1^t.url, \ldots, l_m^t.url\} >$$

where, for $1 \leq k < j \leq m$, $l_k^t \in L$, $l_k^t.ip = ip_t$, $l_k^t.uid = uid_t$, and

$$l_j^t.time - l_k^t.time \leq \text{max time gap}.$$

As an example consider the log entries of Table 1. If the user-specified maximum time gap 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 $\{/, /\sim\texttt{adviser}, /\sim\texttt{adviser/csci-faq.html}\}$.

Definition 2 Let T be the set of all association transactions of the form $< ip_t, uid_t, URL_t >$ (as defined above), where $URL_t = \{l_1^t.url, \ldots, l_m^t.url\}$. We define the *web space*, WS, associated with the access log as $WS = \bigcup_{t \in T} URL_t$.

Let U be a set of URLs ($U \subseteq WS$). We define the *support count* for U 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.

Definition 3 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 Data and Transaction Model for Sequential Patterns

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, therefore, define a transaction to be a set of all the URL names and their access times for the same client within a user-specified maximum time gap.

Definition 4 A *temporal transaction* t is a triple:

$$t =< ip_t, uid_t, UT_t >$$

where

$$UT_t =< (l_1^t.url, l_1^t.time), \ldots, (l_m^t.url, l_m^t.time) >,$$

such that for $1 \leq k < j \leq m$, $l^t_k \in L$, $l^t_k.ip = ip_t$, $l^t_k.uid = uid_t$, and

$$l^t_j.time - l^t_k.time \leq \text{max time gap}^1.$$

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. Note that for each transaction t, the UT–Set is the set of all URL–time pairs of all the pages accessed by the client within the max time gap. We also define the transaction time for t, denoted by $time(t)$, as:

$$time(t) = \max_{1 \leq i \leq m} l^t_i.time$$

Definition 5 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 $1 \leq i \leq k-1$.

Definition 6 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 7 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:

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

[1]In the case of sequential patterns, max. time gap corresponds to the definition of window-size in [SA96]

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).

We can now define a sequential pattern in the context of Web mining.

Definition 8 Let $S = X \cdot Y$ be a UT–sequence (where $\cdot$ denotes concatenation of two sequences). A *sequential pattern* is an expression of the form $X \stackrel{s,\alpha}{\Longrightarrow} Y$, where the *support* s of the rule $X \stackrel{s,\alpha}{\Longrightarrow} Y$ 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 $X \stackrel{s,\alpha}{\Longrightarrow} Y$ (among all UT–sequences $S = X \cdot Y$), where s is at least a given threshold and α is at least another given threshold.

It should be noted that when the maximum gap size in the definition of a temporal transaction is 0, then the elements of each UT–sequence are simply singletons (corresponding to a single log entry for a client). The more general definition of a temporal transaction provides the flexibility of considering several log entries for a user to be considered within a single transaction. In most applications, however, we would only be interested in sequential pattern made up of single URLs and not sets of URLs. For this reason, in the implementation of the WEBMINER, described in the next section, we take the maximum time gap size for transactions to be 0.

3.4 Clustering and Classification

In a clustering and classification task, the definition of transaction depends on the process. For instance, if the process is defined to be clustering based on individual URL names, a *clustering transaction* can be defined to be the same as that of association rules with infinity as a max time gap. If the process is defined to be clustering based on the discovered patterns, $P = \{p_1, p_2, \ldots, p_l\}$, where p_i is $X_i \Longrightarrow Y_i$, then a clustering transaction can be defined as:

$$t = < ip_t, uid_t, \{F_1(UT_t), F_2(UT_t), \ldots, F_l(UT_t)\} >$$

where

$$UT_t = \{(l_{i_1}.url, l_{i_1}.time), \ldots, (l_{i_m}.url, l_{i_m}.time)\}$$

and for $1 \leq k \leq m$, $l_{i_k} \in L$, $l_{i_k}.ip = ip_t$, $l_{i_k}.uid = uid_t$, and

$$F_i(UT_t) = \begin{cases} 1 & \text{if } UT_t \text{ contains the pattern } p_i, \\ -1 & \text{if } UT_t \text{ contains the pattern } X_i \text{ only,} \\ 0 & \text{otherwise.} \end{cases}$$

In classification task, we add additional attributes provided by user to the attributes defined from the clustering task. These additional attributes come from user registration, online survey forms, and techniques such as "anonymous ticketing" [Inc96]. The classes for the transactions are the clustering determined from the clustering task.

4 Implementation

In this section we describe the implementation of WEBMINER, a Web mining system based on the framework presented in the previous sections. 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 also present our implementation of a query mechanism used in WEBMINER to provide more user control and the basis for an effective user interface. In the near future, the system will be extended to incorporate components for clustering and the discovery of classification rules.

The architecture of WEBMINER was presented in Section 3 (see Figure 1). Before any knowledge discovery takes place from the Web data, the raw server log goes through the *data cleansing* process. The resulting data is then formatted appropriately according to the particular application (association rules and sequential patterns require the input data to be in different formats). Instead of mining for all patterns, the query mechanism, built on top of the basic engine, is used to limit the search to relevant and useful patterns. Details of the query mechanism are given in Section 4.2.

As shown in Figure 1, the initial steps of parsing and cleansing are performed on the raw server data irrespective of the application. After that, the processing depends on the particular Web mining task. The implementation of this phase is discussed in the following sections.

4.1 Algorithms for Web Mining Applications

4.1.1 Association Rules

Based on the data and transaction model for association rules, described in Section 3.2, we group the log entries for a client to define a transaction. However, depending on the application and the user-specified maximum time gap, the URLs accessed by the same client in consecutive visits may fall into different transactions.

Given a Web Server Log, 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. An algorithm for finding all associ-

1. L_1 = { large 1-itemsets} ;
2. **for** (k = 2; $L_{k-1} \neq \phi$; k++) do **begin**
3. C_k = apriori_gen(L_{k-1}); // New Candidates
4. get_count(C_k)
5. $L_k = \{c \in C_k \mid$ c.count $\geq$ minsup$\}$
6. **end**
7. Answer = $\bigcup L_k$

Figure 2: Algorithm Apriori

ation rules, henceforth referred to as the *apriori* algorithm was presented in [AS94]. We have used the same algorithm. Following the same notation, a *large* itemset refers to the itemsets satisfying the minimum support requirement. Figure 2 depicts the algorithm.

The *apriori_gen* function generates potential k-itemsets from the set of large (k-1)-itemsets, and the *get_count* function takes the set of potential large k-itemsets and generates the count for all of them. Then, we gather the itemsets which are above the minimum support. A hash tree has been used for implementing the *get_count* function. For details of this algorithm , please refer to [AS94].

4.1.2 Sequential Patterns

In the implementation of WEBMINER, the time gap defining transactions for sequential patterns, defined in section 3.3, 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.

In the algorithm for mining sequential patterns, all the steps, except *apriori_gen* and *get_count* remain the same. 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 to be difficult [AS94]. For getting around this problem, we have changed the joining criteria itself. Instead of matching the first k-1 elements of two k-itemsets, we match the last k-1 elements of the first k-itemset with the first k-1 elements of the second k-itemset. Moreover, while joining, the large set is traversed from the beginning (since the candidate set need not be sorted). Figures 3 and 4 illustrate the difference between the two joining strategies. In the figures, *Large* and *Candidate* refer to large itemset and candidate itemset, respectively. JOIN is the process of joining a large set with itself to produce a new candidate set, whereas PRUNE ON

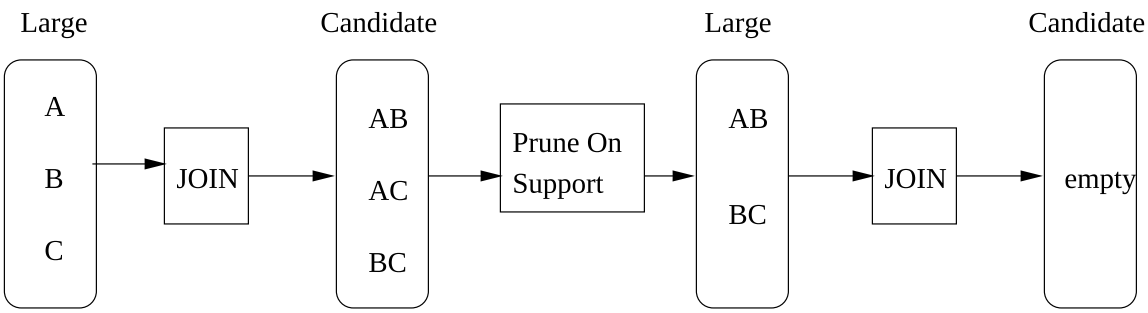

Figure 3: Example of Joining Algorithm for Association Rules

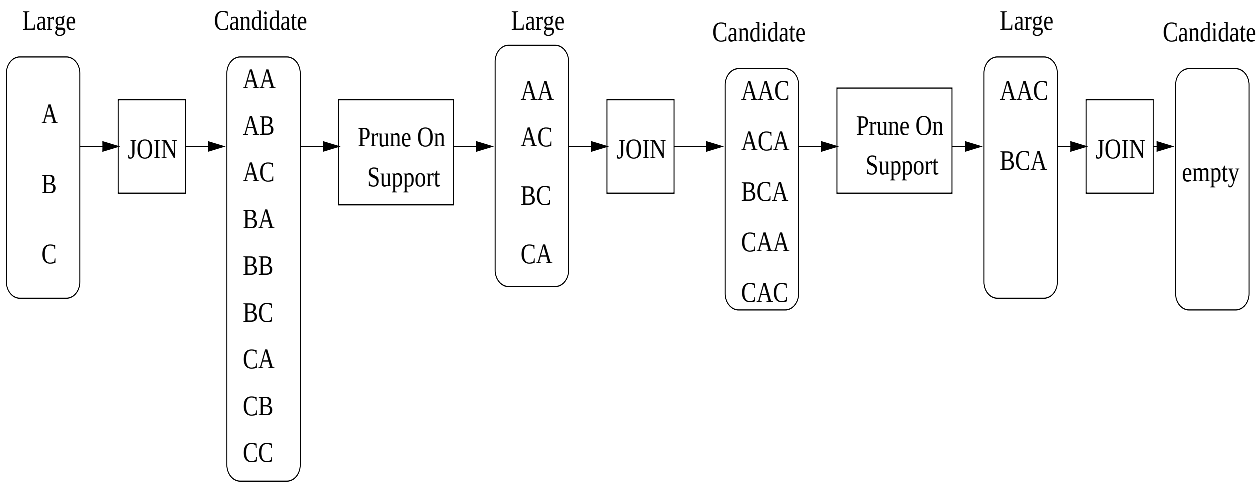

Figure 4: Example of Joining Algorithm for Sequential Patterns

SUPPORT is the process of generating a large itemset from a candidate set by pruning all elements which do not have the user defined minimum support.

For the function *get_count*, the question is: Given a UT-sequence $< A_1, \ldots, A_n >$, how can we determine if there exists a subsequence $< B_1, \ldots, B_k >$ within a certain time range δt? The important factor to consider is that the number of occurrences of that subsequence within the time range is insignificant. The only thing that matters is whether the subsequence is present or not. Figure 5 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 timestamp, a subsequence, and a time interval as input. Here, ISize is the current size of the candidate set, i.e the size of the subsequence given.

```
Initialize prefixtime to -1;
for (all items i in transaction) do begin
   if (firstitem of subsequence matches) then
      prefixtime[1] = transtime[i];
   for ( all other items j in subsequence ) do
      if (item j of subsequence matches) then
         prefixtime[j] = prefixtime [j - 1];
      if (timespan of subsequence ≥ timediff) then
         return TRUE // subsequence found.
end
return FALSE // subsequence not found.
```

Figure 5: Algorithm gen_count

4.2 The Query Mechanism

The need for a query language can hardly be overemphasized. Relational databases owe much of their success to the existence of powerful query language. 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 a SQL-like language which are discussed in more detail below.

4.2.1 Design of the Query Mechanism

In our implementation, we have used a variation of regular expressions to specify constraints on patterns within a data mining query. The following operators provide the basic primitives in our expression language.

1. The unary operator + means one or more instances of any symbol.

2. The unary operator $*$ means zero or more instances of any symbol.

3. The unary operator ? means exactly one instance of any symbol.

These operators can be easily extended, and can be applied to various domains.

For example, if 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 temporal sequences 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 $\geq$ 960101 AND domain = edu AND
support = 1.0 AND confidence = 90.0

4.2.2 Query Optimization Issues

Query Optimization [EN94] has been a traditional problem in all databases. A query typically has many execution strategies, and the problem is to choose a suitable one out of them. The use of query optimization can speedup the process of finding the patterns in many ways. A very simple method to generate only those patterns which the user is interested in is to generate all possible patterns using the original algorithm, and checking in the end which of these patterns match the requirements specified by the user. Another option is to use the query predicates in pattern generation algorithm itself. For example, one should take into consideration the starting and ending times specified in the query, and only consider data within that time range. Having a time index would certainly help in this case. Moreover, the pattern can lead to elimination of some transactions. For example, to get all patterns $A * B * C *$, one does not need to consider a transaction which does not contain all the items in the query, i.e., A, B, and C.

Another way in which the query can be used within the algorithm is by modifying the pruning technique based on the type of the pattern. For example, to search for patterns of the type $A*B*C*$, we need not consider anything not starting with A while generating candidate sets of size two. So, the pruning can be done more efficiently. It can also be used in a variety of other ways. For example, in the above pattern, a group of candidate sets having a common starting symbol can be deleted if none of them contains either of A, B or C. Many of the techniques described above can be used in conjunction with each other. For example, some of the transactions can be pruned first based on the query to generate candidate set. The resulting candidate set can then be pruned further based on specified patterns.

The decision about which method to use depends partly on how the user intends to utilize the discovered knowledge. For one time data mining tasks, it is more efficient to use the query as a constraint in the discovery process. On the other hand, if the user intends to perform different analysis on the same discovered knowledge, then it will be more efficient to store the results of data mining tasks and perform query evaluation on that data.

4.2.3 Implementation of the Query Mechanism

In our implementation, we have used the patterns for data generation and pruning purposes. We chose to work with integers for efficiency. In order to do that, we converted strings (which denote URLs that the different users accessed at various points in time) to integers. We assumed that most of the queries will be on the same data, thereby the conversion of sites to unique integers is a one time process. Therefore, we have not interspersed the two processes (converting strings into integers and removing parts of the redundant data based on the query) and performed them sequentially, although both processes can be executed concurrently. Use of query during data generation is independent of whether data is to be generated for association rules or sequential patterns. In the data elimination phase, we just delete those transactions which do not contain all the items specified in the query. However, pruning effects are dependent on whether you are pruning for association rules or sequential patterns.

5 Experimental Results

We have used the access log of Cray Research Home Page (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. 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.

We experimented with support between 0.05% and 1.0% and confidence of 80%. Table 2 shows the execution time of apriori and rule generation algorithm, the maximum size of the rules

support(%)	apriori time (sec.)	rule generation time (sec.)	max rule size	number of rules
1.0	2.2	0.1	5	22
0.5	2.7	0.2	5	396
0.4	3.7	0.6	6	1806
0.35	4.4	2.6	8	7297
0.3	5.3	10.0	9	28107
0.2	6.9	24.8	11	69507
0.1	8.6	25.0	11	71942
0.05	14.2	45.6	11	130738

Table 2: Association Rules with 80% Confidence.

discovered and the total number of rules discovered. 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. The execution time for our implementation of the apriori algorithm ranges from 2.2 to 14.2 seconds while the execution time for the rule generation ranges from 0.1 to 45.6 seconds. The rule generation algorithm has not been fully optimized and thus it took considerably longer time than apriori algorithm where the rule size was larger than 8.

Table 3 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[2] 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 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[3] home page and T3E top page visit T3E

[2]Trade Mark of Cray Research Inc.
[3]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 3: Some Examples of the Association Rules Discovered

main home page (`CRAY_T3E.html`) about 7% more than other clients who just accessed the T3E main home page.

We have also performed an experiment to find some frequent patterns from the same access log. For this experiment, we did not consider the confidence of the sequential patterns. Table 4 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[4] technical solution for Bayer group and the page containing application vendor quotes, in that order.

Based on the discussion of Section 2, we can already observe how these discovered rules can be useful. For example, the combinations of rules 3 and 4 in Table 3 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]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 4: Some Examples of the Sequential Patterns Discovered

Similarly, the 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 4 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.

6 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 formal data and transaction models for various Web mining tasks (Section 3). We have also described WEBMINER, a system based on the proposed framework (Section 4), and presented experimental results on real-world industrial data to illustrate some of its applications (Section 5).

There are several areas in which our research efforts in Web mining are continuing. Many of these research directions are also relevant to the broader areas of data mining and knowledge discovery in databases.

Currently, we are extending the implementation of WEBMINER to incorporate mechanisms for clustering analysis and discovery of classification rules. One problem of interest is to perform cluster analysis on the association rules and sequential patterns discovered using the methods discussed earlier. After the rules are discovered, users select interesting rules and provide them to the clustering process. The clustering algorithm treats each rule as a basis for an attribute. Each user's transaction, defined in terms of sequential patterns, will be examined to see if it contains the rules. The clusters determined during this process can then be used as the basis for the discovery of classification rules using algorithms such as $C4.5$ [Qui93] on decision trees. The query mechanism in WEBMINER will also be extended to include clustering and classification constraints.

One important area of future work involves the analysis of discovered rules to provide meaningful courses of action to users. It is possible to extend the architecture of the WEBMINER (and possibly other knowledge discovery systems) with a domain specific knowledge-based system to use the discovered rules for providing diagnosis and suggestions to the user (for example about modifying the organization of content within the Web space). For example, suppose we have discovered rules $A, B \Longrightarrow C$ with high confidence, H, and $A \Longrightarrow C$ with low confidence, L. Furthermore, suppose that a user-defined taxonomy specifies that both secondary pages B and C can only be accessed through page A (denoted $B, C < A$). The combination of these facts suggests that there is some portion of the content of B which relates to C, and should be moved up in the hierarchy (to A). More formally, we may specify this "suggestion" as:

$$\frac{A, B \overset{H}{\Longrightarrow} C; \quad A \overset{L}{\Longrightarrow} C; \quad B, C < A}{A \leftarrow A + B_C},$$

where B_C is a portion of the content of B that corresponds to C.

The knowledge-based system can analyze the discovered associations or sequential patterns, and using its rule-base of "suggestions", it can provide diagnosis for restructuring a Web site. Ideas similar to this can be applied to a variety of data mining applications in other domains.

Some other areas of future research are the following:

- Incorporating a user-specified taxonomy on the Web space which, in turn, can be used in pruning during the discovery phase. One important issue here is that often the physical structure of a Web space (i.e., file and directory structure) does not correspond to its logical structure as conceived by the designer or the clients. One possibility is to base the discovery tasks on a pre-specified conceptual taxonomy which is a subset of the actual logical structure of the Web space. This conceptual taxonomy can be used for pruning in the search process and for eliminating useless or redundant rules.

- Incorporating a query processing subsystem on the discovered rules themselves. Currently, the query mechanism is used to specify constraints in the discovery phase. However, a post-discovery query mechanism can be useful in identifying relevant rules by the user from among a potentially large number of discovered rules. The resulting rules from a query can then be saved and used as the basis for further query processing.

- Providing mechanisms for distributed Web mining. It is invariably the case that the Web space for an organization is distributed among many servers with separate access logs. Transmission of all of these access logs to a central location may be very expensive. Thus, it is desirable to be able to perform the discovery tasks on several servers remotely. In the case of association rules or sequential patterns this means the discovery of rules such as $X \Longrightarrow Y$, where X may reside on one server and Y on another server.

- Using client profiles along with discovered rules to do prefetching of pages and client-side caching. For example, a client-side agent, using the information available on the server about the discovered rules and the profile information about the client, may determine that it is likely for the client to access a page p in the same visit. The page p can then be prefetched from the server to provide faster access by the client browser.

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.

[CS96] P. Cheeseman and J. Stutz. Baysian classification (autoclass): Theory and results. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI/MIT Press, 1996.

[EN94] R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. Benjamin/Cummings, Redwood City, CA, second edition, 1994.

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

[Fis95] D. Fisher. Optimization and simplification of hierarchical clusterings. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 118–123, Montreal, Quebec, 1995.

[FPSM91] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases: An overview. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 1–27. The AAAI Press, Menlo Park, CA, 1991.

[FPSS96] U.M. Fayyad, G. Piatetski-Shapiro, and P. Smith. From data mining to knowledge discovery: An overview. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI/MIT Press, 1996.

[HCC93] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. In *IEEE Transactions on Knowledge and Data Eng.*, volume 5, pages 29–40, 1993.

[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.

[HKS+96] E.H. Han, V. Kumar, S. Shekhar, M. Ganesh, and J. Srivastava. Search framework for mining classification decision trees. Technical Report TR-96-023, Department of Computer Science, University of Minnesota, Minneapolis, 1996.

[HS95] M. A. W. Houtsma and A. N. Swami. Set-oriented mining for association rules in relational databases. In *Proc. of the 11th Int'l Conf. on Data Eng.*, pages 25–33, Taipei, Taiwan, 1995.

[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.

[KR90] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis.* John Wiley & Sons, 1990.

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

[MAR96] 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.

[MCPS93] C. J. Matheus, P. K. Chan, and G. Piatetsky-Shapiro. Systems for knowledge discovery in databases. *IEEE Transactions on Knowledge and Data Eng.*, 5(6):903–913, December 1993.

[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.

[NH94] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *Proc. of the 20th VLDB Conference*, pages 144–155, Santiago, Chile, 1994.

[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.

[SA95] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proc. of the 21th VLDB Conference*, pages 407–419, Zurich, Switzerland, 1995.

[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.

[SON95] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. of the 21th VLDB Conference*, pages 432–443, Zurich, Switzerland, 1995.

[WK91] S.M. Weiss and C. A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems.* Morgan Kaufmann, San Mateo, CA, 1991.

[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.