Grouping Web Page References into Transactions for Mining World Wide Web Browsing Patterns *

R. Cooley, B. Mobasher, and J. Srivastava
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455, USA

Abstract

Web-based organizations often generate and collect large volumes of data in their daily operations. Analyzing such data involves the discovery of meaningful relationships from a large collection of primarily unstructured data, often stored in Web server access logs. While traditional domains for data mining, such as point of sale databases, have naturally defined transactions, there is no convenient method of clustering web references into transactions. This paper identifies a model of user browsing behavior that separates web page references into those made for navigation purposes and those for information content purposes. A transaction identification method based on the browsing model is defined and successfully tested against other methods, such as the maximal forward reference algorithm proposed in [1]. Transactions identified by the proposed methods are used to discover association rules from real world data using the WEBMINER system [7].

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 [2, 3, 8] 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. Web server access logs have been used as a testbed for the application of certain data mining tasks such as the discovery of frequent episodes [6]. Recently, *maximal forward references* have been proposed [1] as a way to extract meaningful user access sequences.

The overall task of *Web usage* mining 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 *preprocessing 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.

This paper presents a general model for identifying transactions for data mining from WWW log data. The specific contributions include (i) definition of generic transaction identification modules, (ii) definition of a user browsing behavior model that can be used to separate important or information *content* page references from references used for *navigation* purposes, (iii) development of specific transaction identification modules based on web page reference lengths or web site structure, (iv) and evaluation

This work was supported in part by NSF grant ASC-9554517

0-8186-8230-2/97 $10.00 © 1997 IEEE

of the different transaction identification modules using generated server log data with known association rules.

2 User Browsing Behavior Model

In order to group individual web page references into meaningful transactions for data mining, an underlying model of the user's browsing behavior is needed. A reasonable assumption seems to be that a given user treats each page in one of two ways, either for *navigation* purposes to find links to the desired data, or for actual information or *content* purposes. What is merely a *navigation* page for one user may be a *content* page for another. While it may be possible to classify some web pages based on the number of links versus text contained on the page, most pages will probably not fit clearly into only one category. For example, a page that contains a title followed by a list of links to other pages can easily be classified as a *navigation* page. However, a page with a mix of text and links can not easily be classified as *navigational* or *content*, and could serve a different purpose for different users.

Using the concept of *navigation* and *content* page references, there are two ways to define transactions. The first would be to define a transaction as all of the *navigation* references up to and including each *content* reference for a given user. Mining these *navigation-content* transactions would essentially give the common traversal paths through the web site to a given *content* page. The second method would be to define a transaction as all of the *content* references for a given user. Mining these *content-only* transactions would give associations between the *content* pages of a site, without any information as to the path taken between the pages. It is important to note that results generated from *content-only* transactions only apply when those pages are used as *content* references. For example, an association rule, `A` $\Longrightarrow$ `B`, is normally taken to mean that when `A` is in a set, so is `B`. However, if this rule has been generated with *content-only* transactions, it has a more specific meaning, namely that `A` implies `B` only when both `A` and `B` are used as content references. This property allows data mining on *content-only* transactions to produce rules that might be missed by including all of the page references in a log. If users that treat page `A` as a *navigation* page do not generally go on to page `B`, inclusion of the *navigation* references into the data mining process would reduce the confidence of the rule `A` $\Longrightarrow$ `B`, possibly to the point where it is not reported. Depending on the goals of the analysis, this can be seen as an advantage or a disadvantage. The key is that *navigation-content* transactions can be used when this property is undesirable. The challenge of identifying transactions is to dynamically determine which references in a server log are *navigational* and which are *content.* Even sorting the server log by user identification is not a straightforward task due to anonymous logins and proxy servers. A discussion of the user identification problem is given in [9]. This paper assumes that the page references in a server log can be readily sorted by user identification. Once a log has been sorted by user identification, the date/time stamp and the URL of the page accessed are two other pieces of information in the *Common Log Format* specified as part of the HTTP protocol by CERN and NCSA [5] that can be used to classify each reference in the server log as a *navigation* or *content* reference. Section 3 describes methods to use these pieces of information to classify the references in a web server log.

3 Transaction Identification

3.1 General Model

The raw server log can be thought of in two ways; either as a single transaction of many page references, or a set of many transactions each consisting of a single page reference. The goal of transaction identification is to create meaningful clusters of references for each user. Therefore, the task of identifying transactions is one of either *dividing* a large transaction into multiple smaller ones or *merging* small transactions into fewer larger ones. This process can be extended into multiple steps of *merge* or *divide* in order to create transactions appropriate for a given data mining task. A transaction identification module can be defined as either a *merge* or a *divide* module. Both types of modules take a transaction list and possibly some parameters as input, and output a transaction list that has been operated on by the function in the module in the same format as the input. The requirement that the input and output transaction format match allows any number of modules to be combined in any order, as the data analyst sees fit. 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, however these fields are not used in the transaction model.

Definition 1 A *General Transaction* t is a triple:

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

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

Note that definition 1 assumes that each transaction is made up of references from only one user. Although there may be reasons to create transactions of references from different users, this paper assumes that the initial step in identifying transactions will always be to use a *merge* module based on user id. Once the appropriate transaction identification modules have been applied to the server log, a final preparation module can be used to properly format the transactions for the type of data mining to be accomplished. For example, since temporal information is not needed for the mining of association rules, a final association rule preparation module would strip out the time for each reference, and do any other formatting of the data necessary for the specific data mining algorithm to be used.

3.2 Specific Transaction Identification Modules

Since the initial merge module identifies transactions consisting of all of the page references for a given user, the next step in the transaction identification process will always be the application of a *divide* module. This section describes three *divide* transaction identification modules. The first two, *reference length* and *maximal forward reference*, make an attempt to identify transactions based on the model described in section 2. The third, *time window*, is not based on the section 2 model, and is mainly used as a benchmark to compare with the other two algorithms. All of the modules operate on the not necessarily valid assumption that the sequence of web page references contained in a server log are an accurate representation of the users behavior. Mechanisms such as local caches and proxy servers can severely distort the overall picture of user traversals through a web site. Cache busting and mandatory user registration are two examples of attempts to obtain accurate data, but do not address all of the potential problems. A more complete discussion of the shortcomings of the current log standard and potential solutions are the subject of [9]. The transaction identification modules and experiments presented in this paper do not attempt to address problems surrounding accurate data collection.

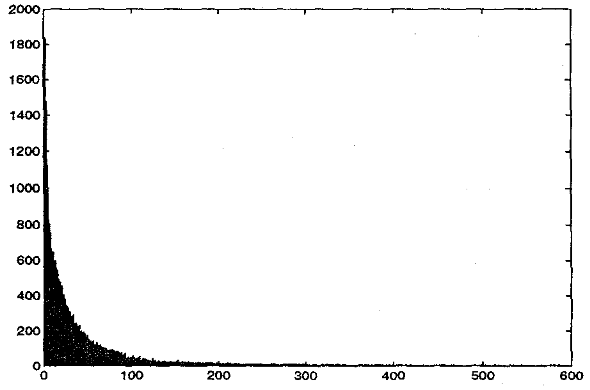

Figure 1: Histogram of Web Page Reference Lengths (seconds)

3.2.1 Reference Length Module. The *reference length* transaction identification module is based on the assumption that the amount of time a user spends on a page correlates to whether the page should be classified as a *navigation* or *content* page for that user. Figure 1 shows a histogram of the lengths of page references between 0 and 600 seconds for a server log from Global Reach Internet Productions (GRIP). Analysis of several other server logs reveals that like figure 1, the shape of the histogram is usually close to an exponential distribution. It is expected that the variance of the times spent on the *navigation* pages is small and the *navigation* references make up the lower end of the curve. The length of the *content* references are expected to have a wide variance and would make up the upper tail that extends out to the longest reference. If an assumption is made about the percentage of *navigation* references in a log, a reference length can be calculated that estimates the cutoff between *navigation* and *content* references. Specifically, given a percent of *navigation* references n, the *reference length* module uses a normal estimate of a chi-squared distribution to calculate the time length that n percent of the references in the log are less than with 95% confidence. The definition of a transaction within the *reference length* module is given in definition 2. It has the same structure as definition 1 with the reference length added for each page.

Definition 2 A *Reference Length Transaction* t_{rl} is a quadruple:

$$t_{rl} = < ip_{t_{rl}}, uid_{t_{rl}}, \{(l_1^{t_{rl}}.url, l_1^{t_{rl}}.time, l_1^{t_{rl}}.length), \ldots,$$

$$(l_m^{t_{rl}}.url, l_m^{t_{rl}}.time, l_m^{t_{rl}}.length)\} >$$

where, for $1 \leq k \leq m$, $l_k^{t_{rl}} \in L$, $l_k^{t_{rl}}.ip = ip_{t_{rl}}$, $l_k^{t_{rl}}.uid = uid_{t_{rl}}$

The length of each reference is estimated by taking the difference between the time of the next reference and the current reference. Obviously, the last reference in each transaction has no "next" time to use in estimating the reference length. The *reference length* module makes the assumption that all of the last references are *content* references, and ignores them while calculating the cutoff time. This assumption can introduce errors if a specific *navigation* page is commonly used as the exit point for a web site. When a log spans a significant portion of time, there is a good chance that the same users will visit the site more than once. In this case, the length of the last reference for the previous visit can be calculated as being hours long. It is highly unlikely that the user actually actively used the web page for this period of time. In order to prevent these outliers from skewing the cutoff time estimation, only the reference lengths of one hour or less are taken into consideration when calculating the cutoff time. While interruptions such as a phone call or lunch break can result in the erroneous classification of a *navigation* reference as a *content* reference, it is unlikely that the error will occur on a regular basis for the same page. A reasonable minimum support threshold during the application of a data mining algorithm would be expected to weed out these errors.

Once the cutoff time is calculated, the two types of transactions discussed in section 2 can be formed by comparing each reference length against the cutoff time. Depending on the goal of the analysis, the *navigation-content* transactions or the *content-only* transactions can be identified. If C is the cutoff time, for *navigation-content* transactions the conditions,

$$\text{for} \quad 1 \leq k \leq (m-1) : l_k^{t_{rl}}.length \leq C$$

$$\text{and} \quad k = m : l_k^{t_{rl}}.length > C$$

are added to definition 2, and for *content-only* transactions, the condition,

$$\text{for} \quad 1 \leq k \leq m : l_k^{t_{rl}}.length > C$$

is added to definition 2. The one parameter that the *reference length* module requires is an estimation of the overall percentage of references that are *navigational.* The estimation of the percentage of *navigation* references can be based on the structure and content of the site or experience of the data analyst with other server logs. The results presented in section 6 show that the module is fairly robust and a wide range of *navigation* percentages will yield reasonable sets of association rules.

3.2.2 Maximal Forward Reference Module. The *maximal forward reference* transaction identification module is based on the work presented in [1]. Instead of time spent on a page, each transaction is defined to be the set of pages in the path from the first page in the log for a user up the the page before a backward reference is made. A new transaction is started when the next forward reference is made. A forward reference is defined to be a page not already in the set of pages for the current transaction. Similarly, a backward reference is defined to be a page that is already contained in the set of pages for the current transaction. The underlying model for this module is that the maximal forward reference pages are the *content* pages and the pages leading up to the each maximal forward reference are the *navigation* pages. For example, an access sequence of `A B C D C B E F E G` would be broken into three transactions of `A B C D`, `A B E F`, and `A B E G`. The *content* pages in this example would be `D`, `F`, and `G`. Like the *reference length* module, two sets of transactions, *navigation-content* or *content-only* can be formed. The definition of a general transaction (definition 1) is used within the *maximal forward reference* module. The *maximal forward reference* module has an advantage over the *reference length* and *time window* modules in that it does not require an input parameter that is based on an assumption about the characteristics of a particular set of data.

3.2.3 Time Window Module. The *time window* transaction identification module simply divides the log for a user up into time intervals no larger than a specified parameter. The module does not try to identify transactions based on the model of section 2, but instead assumes that meaningful transactions have an overall average length associated with them. For a sufficiently large specified time window, each transaction will contain all of the references for each user. Since the *time window* module is not based on the section 2 model, it is not possible to create two separate sets of transactions. The last reference of each transaction does not correspond to a *content* reference, the way it does for the *navigation-content* transactions of the *reference length* and *maximal forward reference* modules. If W is the length of the time window, definition 1 applies for transactions within the *time window* module with the following added condition:

$$(l_m^t.time - l_1^t.time) \leq W$$

Since there is some standard deviation associated with the length of each "real" transaction, it is unlikely that a fixed time window will break a log up appropriately. However, the *time window* transaction module can also be used as a *merge* module in conjunction with one of the other *divide* modules. For example, after applying the *reference length* module, a *merge time window* module with a 10 minute input parameter could be used to ensure that each transaction has some minimum overall length.

4 The WEBMINER System

The WEBMINER system [7] divides the Web usage 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 2.

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 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. The process of removing redundant or irrelevant entries from the Web server log files is referred to as *data cleaning.* A very simple form of 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 using one or a series of transaction identification modules as discussed in section 3.

As depicted in Figure 2, access log data may not be the only source of data for the Web usage 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, various data integration issues are being explored in the context of Web usage mining. For a study of data integration in databases see [4].

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. Finally, a query mechanism will allow the user (analyst) to provide more control over the discovery process by specifying various constraints. The information from the query is used to reduce the scope, and thus the cost of the mining process.

5 Creation of Test Server Log Data

In order to compare the performance of the transaction identification modules presented in section 3 for the mining of association rules, a server log with known rules is needed. Mining of association rules from actual web server logs naturally results in different lists of rules for each module, and even for the same module with different input parameters. It was decided to create server logs with generated data for the purpose of comparing the three modules. The model used to create the data is the user browsing behavior model presented in section 2. The data generator takes a file with a description of a web site as a directed tree or graph and some embedded association rules. The embedded association rules become the "interesting" rules that are checked for during experiments with different transaction identification modules. For each "user", the next log entry is one of three choices, a forward reference, backward reference, or exit. The probability of each choice is taken from the input file, and a random number generator is used to make the decision. If the page reference is going to be a *content* reference, the time is calculated using a normal distribution and a mean time for the page taken from the input file. The times for *navigation* hits are calculated with an exponential distribution. The point of using an exponential distribution for the *navigation* references and a normal distribution with different averages for the *content* references is to create an overall distribution that is similar to those seen in real server logs. Figure 3 shows a histogram of the reference lengths for a generated data set, which is very similar to the histogram of the real server log data shown in Figure 1.

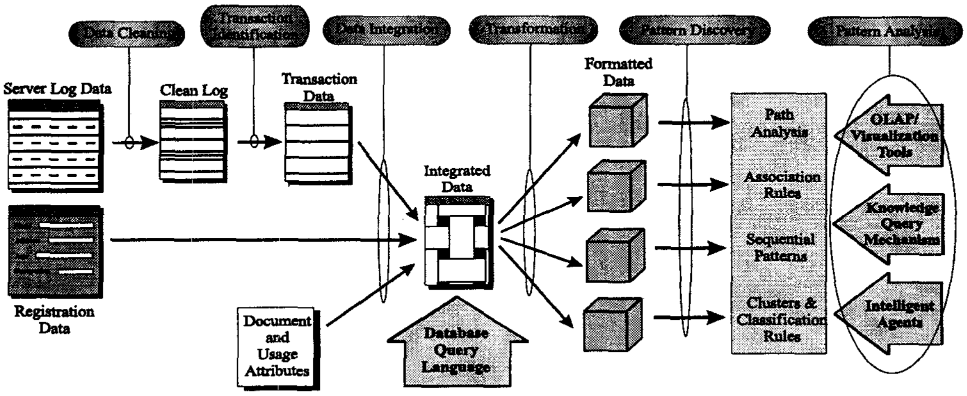

Figure 2: General Architecture for WEBMINER

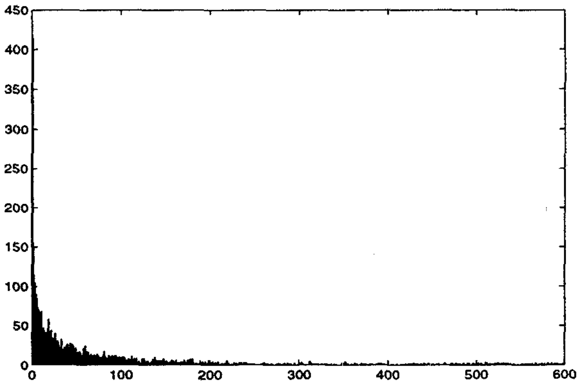

Figure 3: Histogram of Generated Data Reference Lengths (seconds)

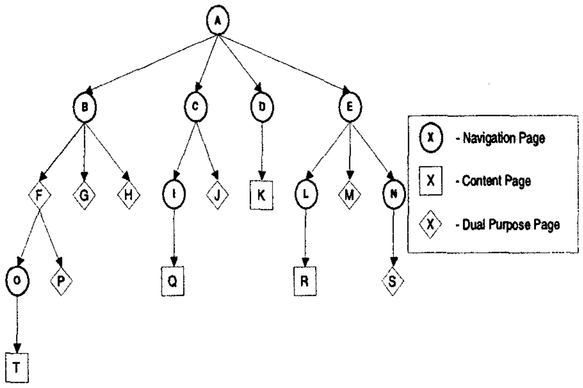

Figure 4: Sparsely Connected Web Site

Besides prior knowledge of the "interesting" association rules, the other advantage of using the generated data for testing the transaction identification modules is that the actual percentage of *navigation* references is also known. The obvious disadvantage is that it is after all, only manufactured data and should not be used as the only tool to evaluate the transaction identification modules. Since the data is created using the user behavior model of section 2, it is expected that the two transaction identification modules based on the same model will perform well.

Three different types of web sites were modeled for evaluation, a sparsely connected graph, a densely connected graph, and a graph with a medium amount of connectivity. The sparse graph, with an average incoming order of one for the nodes, is shown in figure 4. The medium and dense graphs use the same nodes as the sparse graph, but have more edges added for an average order of four and eight respectively.

6 Experimental Evaluation

6.1 Module Comparison using Created Data

Table 1 shows the results of using the three transactions identification modules discussed in section 3 to mine for association rules from data created from the three different web site models discussed in section 5. As expected, the *reference length* module performed the best, since it is based on the same model that is used to create the data. The *maximal forward reference* module performs well for the sparse data, but as the connectivity of the graph increases, its performance degrades. This is because as more forward paths become available, a *content* reference is less likely to be the "maximal forward reference." For dense graphs, *navigation-content* transactions would probably give better results with the *maximal forward reference* module. The performance of the *time window* module is relatively poor, but as the time window increases, so does the performance.

Module	Parameter	Sparse	Medium	Dense
Time Window	10 min.	0/4	0/3	0/3
	20 min.	2/4	2/3	1/3
	30 min.	2/4	2/3	2/3
Reference Length	50%	4/4	3/3	3/3
	65%	4/4	3/3	3/3
	80%	4/4	3/3	3/3
M. F. R.		4/4	2/3	1/3

Table 1: Number of Interesting Rules Discovered

Module	Parameter	Sparse	Medium	Dense
Time Window	10 min.	null	null	null
	20 min.	0.82	0.87	0.87
	30 min.	0.98	0.90	0.88
Reference Length	50%	0.99	0.95	0.96
	65%	1.0	0.99	0.96
	80%	0.97	0.99	0.96
M. F. R.		0.79	0.47	0.44

Table 2: Ratio of Reported Confidence to Actual Confidence

Table 2 shows the average ratio of reported confidence to actual confidence for the interesting rules discovered. The differences between the *reference length* and *maximal forward reference* modules stand out in table 2. The reported confidence of rules discovered by the *reference length* module are consistently close to the actual values. Note that even though the created data has an actual *navigation* page ratio of 70%, inputs of 50% and 80% produce reasonable results. The reported confidence for the rules discovered by the *maximal forward reference* module is significantly lower than the actual confidence, and similar to the results of table 1, degrades as the connectivity of the graph increases.

Table 3 shows the running time of each transaction identification module, and the total run time of the data mining process. The total run times do not include data cleaning since the data was generated in a clean format. Although the data cleaning step for real data can comprise a significant portion of the total run time, it generally only needs to be performed once for a given set of data. The *time window* module shows the fastest module run time, but a much slower overall data mining process due to the number of rules discovered. The *reference length* module has the slowest module times due to an extra set of file read/writes in order to calculate the cutoff time. All three of the modules are O(n) algorithms and are therefore linearly scalable.

Module	Parameter	Sparse	Medium	Dense
Time Window	10 min.	0.81/4.38	0.82/4.65	0.75/3.94
	20 min.	0.84/7.06	0.80/7.06	0.73/4.42
	30 min.	0.79/7.16	0.77/9.95	0.72/5.17
Ref. Length	50%	1.82/4.62	1.66/4.46	1.47/4.09
	65%	1.68/4.29	1.72/4.35	1.45/4.02
	80%	1.62/4.14	1.66/4.26	1.48/4.03
M. F. R.		1.26/3.98	1.30/3.95	1.20/3.87

Table 3: Module Run Time (sec) / Total Run Time (sec)

6.2 Association Rules from Real Data

Transactions identified with the *reference length* and *maximal forward reference* modules were used to mine for association rules from a GRIP server log. The server log used contained 20.3 Mb of raw data, which when cleaned corresponded to about 51.7K references. Because the GRIP web server hosts web sites for other companies, the server log is really a collection of smaller server logs and the overall support for most discovered association rules is low. Accordingly, the association rule generation algorithm was run with thresholds of 0.1% support and 20% confidence. This led to a fairly large number of computed rules (1150 for the *reference length* module and 398 for the *maximal forward reference* module). Table 4 shows some examples of association rules discovered.

The first two rules shown in table 4 are straight forward association rules that could have been predicted by looking at the structure of the web site. However the third rule shows an unexpected association between a page of the *cyprus-online* site and a page from the *MTI* site. Approximately one fourth of the users visiting `/cyprus-online/dailynews.htm` also chose to visit `/mti/Q&A.htm`. Since the *cyprus-online* page no longer exists on the GRIP server, it is not clear if the association is the result of an advertisement, a link to the *MTI* site, or some other factor. The fourth rule listed in table 4 is one of the 150 rules that the *maximal forward reference* module discovered that was not discovered by the *reference length* module. While the *reference length* module discovered many rules involving the *MTI* web site, the *maximal forward reference* module discovered relatively few rules involving the *MTI* site. An inspection of the *MTI* site revealed that the site is a fully connected graph. Consistent with the results of section 6.1, the *maximal forward reference* module does not perform well under these conditions. The association rule algorithm was run with *navigation-content* transactions created from the *maximal forward reference* module to confirm the theory

Module Used	Confidence(%)	Support(%)	Association Rules
Reference Length (content-only)	61.54	0.18	`/mti/clinres.htm /mti/new.htm` ⟹ `/mti/prodinfo.htm`
Reference Length (content-only)	100.00	0.15	`/mti/Q&A.htm /mti/prodinfo.htm /mti/pubs.htm` ⟹ `/mti/clinres.htm`
Reference Length (content-only)	26.09	0.14	`/cyprus-online/dailynews.htm` ⟹ `/mti/Q&A.htm`
Maximal Forward Reference (content-only)	52.17	0.14	`/cyprus-online/Magazines.htm` `/cyprus-online/Radio.htm` ⟹ `/cyprus-online/News.htm`
Maximal Forward Reference (nav-content)	73.50	1.32	`/mti/clinres.htm /mti/new.htm` ⟹ `/mti/prodinfo.htm`

Table 4: Examples of Association Rules from www.global-reach.com

that the rules missed by the *content-only* transactions would be discovered. The last rule listed in table 4 is the same as the first rule listed, and shows that the *navigation-content* transactions from the *maximal forward reference* module can discover rules in a highly connected graph. However, at thresholds of 0.1% support and 20% confidence, approximately 25,000 other rules were also discovered with the *navigation-content* module.

7 Conclusions

This paper has presented a general model for transaction identification for Web usage mining, the application of data mining and knowledge discovery techniques to WWW server access logs. This paper also presented experimental results on created data for the purpose of comparing transaction identification modules, and on real-world industrial data to illustrate some of its applications (Section 6). The transactions identified with the *reference length* module performed consistently well on both the real data and the created data. For the real data, only the *reference length* transactions discovered rules that could not be reasonably inferred from the structure of the web sites. Since the important page in a traversal path is not always the last, the *content-only* transactions identified with the *maximal forward reference* module did not work well with real data that had a high degree of connectivity. The *navigation-content* transactions led to an overwhelmingly large set of rules, which limits the value of the data mining process. Future work will include further tests to verify the user browsing behavior model discussed in section 2 and tests with transactions created from combinations of *merge* and *divide* modules. An important area of ongoing research is to continue to develop 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.

References

[1] 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 Distributed Computing Systems*, pages 385–392, 1996.

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

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

[4] 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.

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

[6] 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.

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

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

[9] J. Pitkow. In search of reliable usage data on the www. In *Sixth International World Wide Web Conference*, pages 451–463, Santa Clara, CA, 1997.