Distributed and Parallel Databases, 13, 161–180, 2003

On Using a Warehouse to Analyze Web Logs

KARUNA P. JOSHI kjoshi1@cs.umbc.edu
ANUPAM JOSHI joshi@cs.umbc.edu
YELENA YESHA yeyesha@cs.umbc.edu
Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD 21250, USA

Recommended by: Ahmed Elmagarmid

Abstract. Analyzing Web Logs for usage and access trends can not only provide important information to web site developers and administrators, but also help in creating adaptive web sites. While there are many existing tools that generate fixed reports from web logs, they typically do not allow ad-hoc analysis queries. Moreover, such tools cannot discover hidden patterns of access embedded in the access logs. We describe a relational OLAP (ROLAP) approach for creating a web-log warehouse. This is populated both from web logs, as well as the results of mining web logs. We discuss the design criteria that influenced our choice of dimensions, facts and data granularity. A web based ad-hoc tool for analytic queries on the warehouse was developed. We present some of the performance specific experiments that we performed on our warehouse.

Keywords: web mining, data warehouse, web logs, e-commerce, adaptive websites

1. Introduction

Web mining can be viewed as the extraction of structure from an unlabeled, semi-structured data set containing the characteristics of users/information respectively [8]. It encompasses several related areas, including mining the content of web pages, mining the structure of web pages are linked, and mining the usage/access of web pages. Our focus in this paper is on web usage/access mining. The logs kept by web servers provide a classic example of usage/access data that can be mined. Analyzing and exploring regularities in the behavior of the users accessing a web site can improve system performance, enhance the quality and delivery of Internet information services to the end user, and identify population of potential customers for electronic commerce [19]. Many approaches have been suggested to mine information from web access log records collected from servers [6, 12, 14, 16]. Typically data has to be scrubbed and preprocessed as a precursor to mining. Such data can also be stored in a data warehouse and be amenable to OLAP like queries. The objective of this paper is to outline a design used to create a ROALP warehouse from web logs. We also describe the web based analysis tool we have created that helps users in querying the warehouse in an ad-hoc manner.

Of late, new tools promising to apply data warehousing and mining techniques on web logs have entered the market. However, most of these tools do not really build a warehouse, and their notion of mining is limited for the most part to simple aggregate statistical analysis. For the most part, they continue the tradition of utilities such as AccessWatch and http-analyze.

Such tools use Java, Perl scripts or shell scripts, to generate reports about web site utilization. A listing of many such utilities can be found at [1]. Most of these tools generate canned reports about number of Kbytes transferred, access organized by domains etc. Their fixed reporting format does not help in analyzing log data from every perspective and customizing these reports requires essentially rewriting the utility. Moreover, these tools typically cannot discover hidden patterns of access embedded in the access logs. The only exception seems to be IBM's SurfAid (http://surfaid.dfw.ibm.com/) that combines an analysis tool with some data mining capabilities (the web site and white papers for this do not specify in detail their approach used).

Broadly speaking, 'Web Mining' has been defined as applying data mining techniques on web data to discover knowledge. However, as mentioned earlier, the data that is actually mined is varied, and different approaches have been followed. The usage mining approach applies mining techniques on the web logs maintained by the servers so as to discover user access and traversal patterns [5, 6, 18–20]. In this paper, we have primarily concentrated on this research direction. Most of the research in this vein interprets web logs to be a semi-structured source of data that can be analyzed using existing mining techniques. In this approach, the aim is to discover the in-built relationships between the various attributes of the log like URLs, IP addresses etc. For instance, Zaine et al. [19] have mined information from web logs by implementing a MOLAP warehouse containing web log data and mining information off it. However, their system is built on their own custom multidimensional database system. Nasraoui et al. [13] have developed new robust fuzzy clustering algorithms and used them on web logs, where as Zarkesh et al. [20] have developed clustering algorithms for user navigation patterns. Other researchers have interpreted web logs essentially as text files and applied phrasal mining techniques to the logs [3, 11]. Chen et al. [5] have used maximal forward traversal to sessionize logs, and then mine them using the DSS algorithm. This work is available as the SpeedTracer [17] tool. Bruchner and Mulvenna [4] propose a hypercube based model, and its equivalent snowflake/star schemata that combines web logs with e-tailing data for analysis of marketing data from e-commerce. However this work primarily describes the concept, as opposed to detailing implementation.

In this paper, we present the design of a warehouse for web log analysis. Our system is different from much of the prior work in as much as it creates a combined data warehouse for both raw access logs, as well as mined data such as clusters representing user access patterns. As explained in later sections in this paper, this allows analytic queries not just on the access log data, but also on mined access patterns. Moreover, our system is built on top of a COTS relation database system (Oracle), and does not require specialized a Multidimensional DBMS. The system has a web based front end that allows for preformatted queries, as well as the possibilities for ad-hoc querying of the warehoused data. We present the design of the system, the tradeoffs involved via experimental studies, implementation details and experimental results.

2. Preprocessing web logs

As a precursor to creating the warehouse, we need to clean our web log data. The textual format of the log data is not suited for import into the database (Oracle). In our design, we have included each log entry as a potential record for our warehouse. We have also included log entries that resulted in errors or redirects in our analysis. The Web sites whose logs we used as data for our system do not maintain a user profile database and don't have cookies. Hence we have not accounted for these in our system. Some prior work in web log mining assumes that **identd** [5, 6] daemons are running and making available identities of the remote users accessing the site—such an assumption is mostly not true in general web sites. We use the approach suggested by Nasraoui et al. [12] that defines sessions using temporally compact accesses from a given IP address. Joshi and Krishnapuram [9] have shown more recently that certain type of cookies that do not store identifying information can help better sessionize the logs. Our design can be trivially extended to handle cookies for sites that store them. More detailed discussion of sessionizing approaches can be found in [10].

Figure 1 is a pictorial representation of our entire system. Our warehouse consists of an access log part (see figure 2(a) and (b)) that allows users to analyze the web logs; and a mining part (see figure 3) that allows users to analyze traversal patterns. In the preprocessing phase, we have created different files for populating the warehouse.

In designing the access log warehouse, we included each log entry as a record in the warehouse, which was created in an Oracle 8 database using star schema. Before loading log entries into the warehouse, we reformatted them to confirm with the formats that Oracle SQL Loader accepts. This was done using a Perl script we created by modifying Follow [7]. Log structure was assumed to be that prescribed by the Common Log Format, $\{\langle\text{domain}\rangle$ $\langle\text{identity}\rangle$ $\langle\text{authorization}\rangle$ $\langle\text{date}\rangle$ $\langle\text{method}\rangle$ $\langle\text{URL}\rangle$ $\langle\text{protocol}\rangle$ $\langle\text{status}\rangle$ $\langle\text{size returned}\rangle\}$ and any entry not conforming to this structure was discarded. We also disregarded the identity and authorization attributes in the web log since they were not being logged. We have included log entries that resulted in errors or redirects in our analysis.

For the mining part, we filtered out other unwanted entries like record accesses to image files that were embedded in the web pages whose 'hit' had already been logged. For the preprocessing phase, we created a file listing all the sessions (described in the next section)

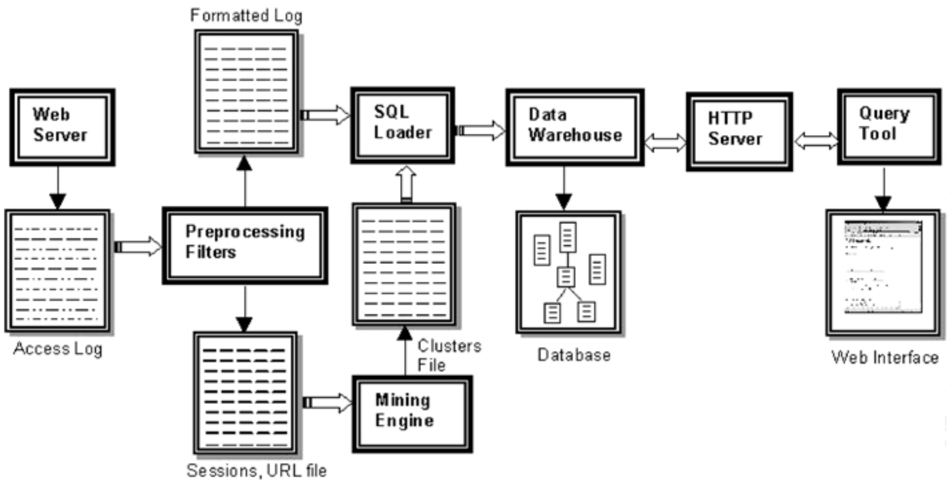


Figure 1. System architecture.

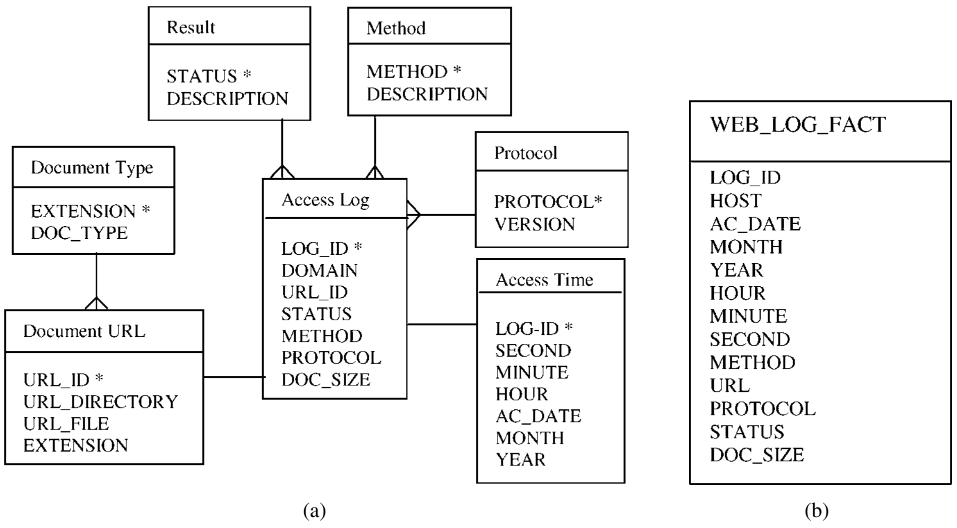


Figure 2. (a) Normalized database for storing web and (b) Web log fact table.

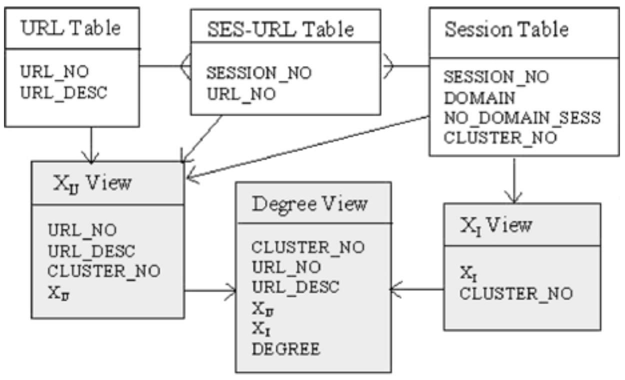


Figure 3. Clustering analysis database structure.

obtained from the logs. The *session* file consists of Session number and the Domain of that session. A URL file was created that contained all the URLs in the web log that was analyzed and a unique number associated with the URL. Once the sessions were clustered as described in later sections, a cluster file listed each session and the cluster that it belongs to. Using the three files as our input, we proceeded to populate the warehouse for the mined data.

3. Warehouse architecture

In a data warehouse Facts defined along with certain attributes (called dimensions) are the core data elements being analyzed. In this system, the web log entries are facts. Given the structure of the web logs, the following are obvious dimension candidates: URL of the page accessed (Document URL), Date of the request (Access Time), method of the request (Method), Result of the request (Result), and Protocol.

The fact tables in the warehouse are directly created from the web logs. The web log data is analogous to the transactional data in an OLTP system and the main fact table was populated from it using Perl scripts we have written. This makes both the creation and maintenance of the warehouse relatively straightforward. This table contained all the log entries that were ever made in our web logs (figure 2(b)). Our web analysis does not build on an existing transactional database, since the web server uses flat files to store its transactional data (accesses) without any normalization. Figure 2(a) illustrates the table structure for a corresponding normalized, relational database. It has "Dimension" tables for data such as Request status codes, request methods, Web protocols etc. The arrows indicate the relationships between dimensions. The dimension data is typically static and dependent on the domain. It also helps map "codes" relating to the HTTP protocol to their descriptions in an efficient manner.

3.1. Performance analysis

While the main fact table can answer any analysis query, aggregation queries where one of the parameter was constructed would take a time. The guiding principle in our architecture was to trade storage space for time. Based on some expected queries, smaller fact tables were created that contain data in a particular range of a dimension. For example, we created separate fact tables for each month of the log. If a user specified the month in the 'date accessed' field, then the fact table corresponding to the month queried would be accessed. Similarly tables were created based on particular status codes, like Error codes (status range 400). Because of their smaller size, these fact tables vastly reduce the time to query the warehouse. Consider a case (Case 1), where we wanted to find out how many hits to our web site were from educational institutions (domain 'edu') for a given month. Querying it from the smaller fact table that stores logs for the desired month is almost 6 times faster than querying it from the main fact table, as seen in Table 1. Consider another case (Case 2) where a web administrator wanted to find out all the links (URLs) in the web

Table 1. Performance measures for the fact tables: Case 1.

Query	Description of the table queried	No. of records in the table	Time taken (ms)	Speedup	Disk space (blocks)
select count(*) from AUG_LOG_FACT where HOST LIKE '%.edu';	Fact table sliced by dimension month	36529	14	5.8	2420
select count(*) from WHOLE_LOG_FACT1 where HOST LIKE '%.edu' and MONTH='Aug';	Main FACT table	242165	82	–	16260

Table 2. Performance measures for the fact tables: Case 2.

Query	Description of the table queried	No. of records in the table	Time taken (ms)	Speedup
select count(*) from JUL_LOG_FACT where STATUS=‘40%’;	Fact table sliced by dimension month—July	43059	17	5.11
select count(*) from ERR_STAT_TBL where MONTH=‘Jul’;	Fact table sliced by dimension status	1468	2	43.5
select count(*) from WHOLE_LOG_FACT1 where STATUS=‘40%’ and MONTH=‘Jul’;	Main FACT table	242165	87	–
select count(*) from AUG_LOG_FACT where STATUS=‘40%’;	Fact table sliced by dimension month—Aug.	36529	18	4.8
select count(*) from ERR_STAT_TBL where MONTH=‘Aug’;	Fact table sliced by dimension status	1468	2	44
select count(*) from WHOLE_LOG_FACT1 where STATUS=‘40%’ and MONTH=‘Aug’;	Main FACT table	242165	88	–
select count(*) from SEP_LOG_FACT where STATUS=‘40%’;	Fact table sliced by dimension month—Sep.	34736	14	5.9
select count(*) from ERR_STAT_TBL where MONTH=‘Sep’;	Fact table sliced by dimension status	1468	2	41
select count(*) from WHOLE_LOG_FACT1 where STATUS=‘40%’ and MONTH=‘Sep’;	Main FACT table	242165	83	–

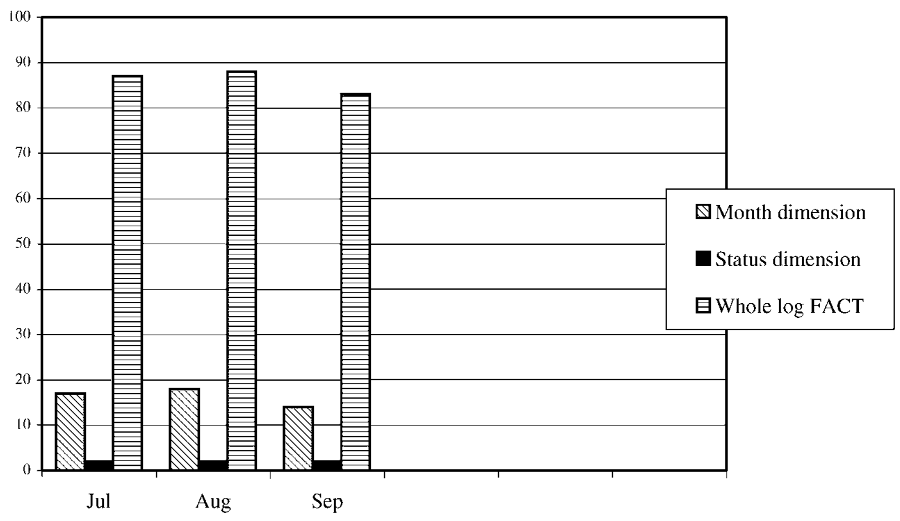


Chart 1. Performance measures for the fact tables: Case 2. Y-axis is time taken in ms.

site that resulted in error state for a given month. S/he could query it either from the main fact table or from the fact table that stores the desired month's data or from the fact table that contains only error code logs. Table 2 and Chart 1 display the performance results that were obtained when the same generic query was executed on the different Fact tables. We observe that fact tables that were created by slicing on a particular dimension (like MONTH or STATUS) did perform better since they contained fewer records. The query executed almost six times faster when run on smaller tables. To analyze this further, we populated the Main FACT table with varying number of records—records taken for each month and also created corresponding fact tables by slicing on the STATUS dimension. Table 3 and Chart 2 below illustrate this example in detail. We observe that the performance improvement is similar in all the cases strongly pointing that STATUS dimension is a good candidate for smaller fact tables. So it is definitely advantageous if similar fact tables are created in the warehouse to generate reports that have to be run frequently. However on the negative side, such tables take up additional space in the database and they will have to be recreated/populated every time data is added to the warehouse (main fact table). Additionally, the main fact table allows more ad-hoc querying capability and better comparative analysis than the smaller fact tables since it contains all the data. Table 4 tabulates the additional blocks that were required to store the smaller fact tables. We observe that tables created for STATUS dimension required almost no extra space, and also improved the response time drastically.

We also observed that every dimension was not a good candidate for creating smaller tables. For example, the dimension METHOD had three main entries of GET, POST and HEAD in our main fact table, where logs with 'GET' method accounted for almost 99% of the records. In such an instance, a smaller FACT table for method 'GET' will provide no appreciable improvement over querying the main FACT table.

Table 3. Performance measures for the fact tables: Case 3.

No.	Description of the table queried	No. of records in the table	Time taken (ms)
1	Whole fact table with 2 months data	170900	140
2	Fact table sliced by dimension status—2 M	19	2
3	Whole fact table with 3 months data	207429	187
4	Fact table sliced by dimension status—3 M	21	3
5	Whole fact table with 4 months data	242165	192
6	Fact table sliced by dimension status—4 M	1468	7

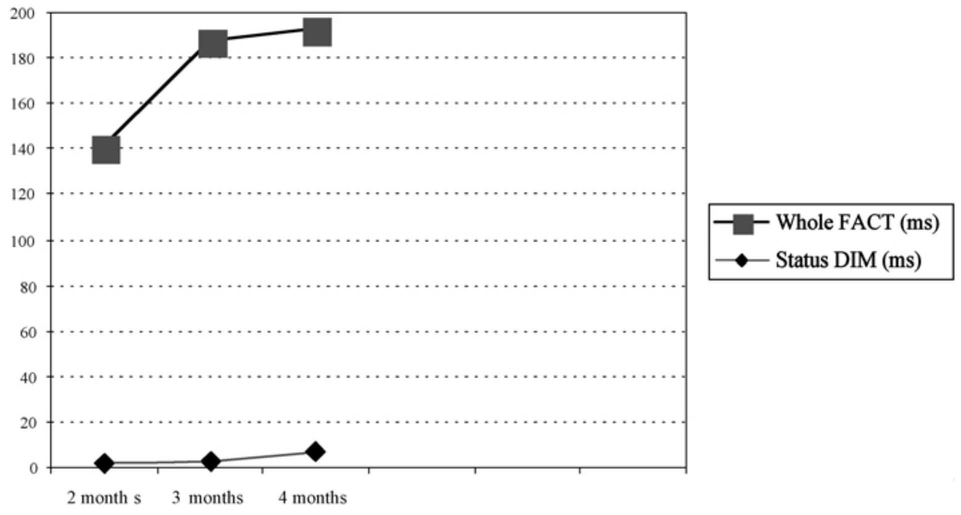


Chart 2. Performance measures for the fact tables: Case 3. Y-axis is time taken in ms.

3.2. Warehouse for analyzing data mining results

In addition to the fact tables, a normalized schema was added to the warehouse to populate the mining results so as to facilitate on line analysis. Figure 3 represents the Entity-Relationship diagram of this database that consists of three tables. The *URL* table stores the URL description along with the Unique ID (URL_NO) that is generated by the program. *Session* Table contains data pertaining to the Domain that identifies the session, the cluster number to which the session belongs and the frequency of the domain. Session-No is the unique identifier for each session. We know that many URLs (say N) can be accessed in a single session and also one URL can belong in multiple sessions (say N). To incorporate the $N : N$ relationship between URLs and Sessions, we split the relationship into a $1 : N$ relation by introducing a new table SES_URL table. This table contains the primary identifiers of both the Session and URL tables and is populated from the output from the clustering experiment.

Table 4. Extra space requirements for the fact tables.

Description of the table queried	No. of records in the table	Disk space (blocks)	Extra space
Fact table sliced by dimension month—July	43059	3010	18.5%
Fact table sliced by dimension month—Aug.	36529	2420	14.9%
Fact table sliced by dimension month—Sep.	34736	2255	13.9%
Fact table sliced by dimension status containing 2 months of data	19	1	~0
Fact table sliced by dimension status containing 3 months of data	21	2	~0
Fact table sliced by dimension status containing 4 months of data	1468	110	0.05% over fact table
Main FACT table containing 2 months of data	170900	11627	–
Main FACT table containing 3 months of data	207429	14055	–
Main FACT table containing 4 months of data	242165	16260	–

We have created three views to help in our analysis. X_i *View* displays the cardinality (total number of sessions) of each cluster obtained. X_{ij} *View* displays each URL in the log along with Cluster number and the total number of sessions in the cluster that contain the URL (displayed by field X_{ij}). The third view, *Degree View*, displays each URL in the log along with Cluster number, X_i value, X_{ij} value and the Degree Measure, X_{ij}/X_i. The Degree measure is the equation for P_{ij} listed in above. The Degree view is the main view that we use for our analysis. It is also the view queried by the Web Interface. An additional view, ALL_VIEW, displays all the main fields in the tables and is also accessed by the Web Interface.

4. Mining web logs

To mine knowledge from the web logs we used two data mining techniques, namely Clustering and Association Rules generation. For the data mining experiments, we collected a variety of logs and ran the association rules generator and clustering algorithm on it. Before mining information from the logs we created sessions from the logs. For discovering association rules, we used SGI's Mineset [15] program that implements a variation of the Apriori algorithm [2]. For Clustering we used the Fuzzy C-medoids algorithm developed by Krishnapuram et al. [18]. This approach allows us to capture a graded (as opposed to binary) notion of similarity between sessions.

4.1. Association rules

To generate the association rules for the web logs, we first generated the sessions and URL files. MineSet requires two input files for generating the association rules—a schema file and a data file. The schema file describes the structure/fields of the data file. The data file consisted of the binary session vectors. Each session can be thought of as a transaction and each URL as an item, in the association rule context. Thus a session [10110] would imply that URLs 1, 3 and 4 were visited in this session and URLs 2 and 5 were not. To generate the association rules for the web logs, we first generated the sessions and URL files. The schema file was simply the URL numbers and was generated using a Perl program. The association file generated by MineSet was cleaned before the results could be analyzed.

4.1.1. Experiment results. We ran the association rules generator for various web logs. Separate schema and data files were generated for each log mined. The output file generated by MineSet was filtered to remove results that were not germane to our study. E.g. association rules that did not have the minimum support (80%) and/or confidence (20%), association rules that stated that if $s_j^{(i)}$ of a URL j was 1, $s_k^{(i)}$ of another URL k was 0 etc. Table 5 below tabulates the association rules result for one of the experiments.

4.1.1.1. [Experiment I] UMBC web page. Support of 80% (and above) and a confidence of 20% and above.

4.1.1.2. Experimental observations. Some observations that were made after studying the generated association rules for UMBC pages are listed below.

1. The Search page is associated with Directory page, Student Link pages, List of Departments page and Summer 99 schedule (schedule of next semester). This indicates that the Search utility was mainly used by new students or students who were new browsers and were not familiar with the web site structure.
2. The URL containing Under Graduate information is associated with Admissions pages, Graduate Program page, Faculty and Academics pages and Schedule pages.
3. Schedule page is associated with Graduate Program page, Under Graduate Program page, Faculty and Academics pages. It is also associated with Summer 99 and Spring 99 schedule pages. This reflects the time when these logs were studied—late in Spring semester, when students were most interested in the current semester and deciding about the one following it.
4. Student Link is associated with the Search page, Directory page, Library and Computing page, Calendar, List of Departments and Schedule pages (Spring and Summer schedules).

4.2. Clustering

Before generating clusters for our sessions, we had to define a measure of similarity between our sessions. The similarity measure that we used for our experiments has been proposed by Nosraoui et al. [13]. They have defined a dissimilarity measure between all session pairs (i.e., the relation matrix) prior to the clustering process. Note that since sessions are not object data, a distance measure in the sense of Minkowski norms is not automatically available.

4.2.1. Evaluating the results. We interpret the results of applying the clustering algorithm on the user session data by using the following quantitative measures. The clustering algorithm [18] assigns user sessions to the closest clusters based on the similarity measures.

Table 5. Association rules for UMBC web page.

URLs		Associated with
{/Search/}	→	{/Directory/}, {/StudentLink/}, {/FacAcademics/Depart/}, {/AboutUMBC/Schedule/summer1999/}
{/AboutUMBC/Schedule/spring1999/}	→	{/StudentLink/}, {/LibComp}, {/UnderGrad}, {/FacAcademics/Depart/}, {/AboutUMBC/Schedule/}, {/AboutUMBC/Schedule/summer1999/}
{/Directory/}	→	{/Search}, {/StudentLink/}, {/Admissions/}, {/FacAcademics/}, {/FacAcademics/departs.html}, {/FacAcademics/depart/}
{/StudentLink/}	→	{/Search}, {/AboutUMBC/Schedule/spring1999/}, {/Directory/}, {/LibComp/}, {/Admissions/}, {/calendar}, {/FacAcademics/Depart/}, {/AboutUMBC/Schedule/}, {/AboutUMBC/Schedule/summer1999/}
{/LibComp/}	→	{/AboutUMBC/Schedule/spring1999/}, {/StudentLink}, {/calendar}, {/AboutUMBC/Schedule/summer1999/}
{/Admissions/}	→	{/Directory/}, {/StudentLink/}, {/Admissions/undergrad.html}, {/GradProg/}, {/UnderGrad/}, {/FacAcademics/departs.html}, {/FacAcademics/Depart/}, {/AboutUMBC/Schedule/}, {/AboutUMBC/Schedule/summer1999/}
{/calendar}	→	{/AboutUMBC/Schedule/spring1999/}, {/StudentLink/}, {/AboutUMBC/Schedule/}, {/Admissions/undergrad.html}, {/LibComp}, {/AboutUMBC/Schedule/summer1999/},
{/Admissions/undergrad.html}	→	{/Admissions/}, {/calendar}, {/GradProg/}, {/UnderGrad},{/AboutUMBC/Schedule/summer1999/}
{/GradProg/}	→	{/Admissions/}, {/Admissions/undergrad.html}, {/UnderGrad/}, {/FacAcademics/}, {/FacAcademics/departs.html}, {/FacAcademics/Depart/}, {/AboutUMBC/Schedule/}
{/FacAcademics/}	→	{/Directory/}, {/GradProg/}, {/AboutUMBC/Schedule/}, {/AboutUMBC/Schedule/summer1999/}, {/UnderGrad/}, {/FacAcademics/departs.html}, {/FacAcademics/Depart/}
{/FacAcademics/departs.html}	→	{/Directory/}, {/Admissions/}, {/GradProg/}, {/UnderGrad/}, {/FacAcademics/}, {/AboutUMBC/Schedule/}, {/FacAcademics/Depart/}
{/UnderGrad/}	→	{/AboutUMBC/Schedule/spring1999/}, {/Admissions/}, {/Admissions/undergrad.html}, {/GradProg/}, {/FacAcademics/},{/FacAcademics/departs.html}, {/FacAcademics/Depart/}, {/AboutUMBC/Schedule/}, {/AboutUMBC/Schedule/summer1999/}
{/FacAcademics/Depart/}	→	{/Search}, {/AboutUMBC/Schedule/spring1999/}, {/Directory/}, {/StudentLink/}, {/Admissions/}, {/GradProg/}, {/UnderGrad/}, {/FacAcademics/}, {/FacAcademics/departs.html}, {/AboutUMBC/Schedule/}, {/AboutUMBC/Schedule/summer1999/}
{/AboutUMBC/Schedule/}	→	{/AboutUMBC/Schedule/spring1999/}, {/StudentLink/}, {/Admissions/}, {/calendar}, {/GradProg/}, {/UnderGrad/}, {/AboutUMBC/Schedule/summer1999/}, {/FacAcademics/}, {/FacAcademics/departs.html}, {/FacAcademics/Depart/},
{/AboutUMBC/Schedule/summer1999/}	→	{/Search}, {/AboutUMBC/Schedule/spring1999/}, {/StudentLink/}, {/LibComp/}, {/Admissions/}, {/calendar}, {/UnderGrad/}, {/FacAcademics/},{/FacAcademics/Depart/}, {/AboutUMBC/Schedule/}, {/Admissions/undergrad.html},

This creates C clusters $X_i = \{s^{(k)} \in S \mid d_{ik} < d_{jk}\ \forall j \neq i\}$, for $1 \leq i \leq C$. The sessions in cluster X_i are summarized in a typical session "profile" vector $P_i = (P_{i1}, \ldots, P_{i|U|})^t$. The components of P_i represent the *degree* with which each URL belongs to a session. This is computed as

$$D_{ij} = p\big(s_j^{(k)} = 1 \,\big|\, s^{(k)} \in X_i\big) = |X_{ij}|/|X_i| \quad \text{where } X_{ij} = \big\{s^{(k)} \in X_i \,\big|\, s_j^{(k)} > 0\big\}.$$

In other words, it measures the fraction of the clusters in the session which included this URL. This helps us recognize those URL which form the "core" of the profile. For some URLs the degree is greater than 1. This is because we have grouped together all the URLs that belong to the same directory. Besides summarizing profiles, the components of the profile vector can be used to recognize an invalid profile, which has no strong or frequent access pattern.

4.2.2. Experiment results. We generated clusters for various logs. While analyzing our clusters, we did not consider clusters that had less than 3 user sessions. In all the clusters $\{/\langle\text{url}\rangle\}$ and $\{/\langle\text{url}\rangle/\}$ were regarded the same and counted only once. Separate tables were created in the Oracle database for each log studied. The tables and views had similar structure as described in above. The *Degree View* for each log was used in analyzing the results. Table 6 below tabulates the clusters that were found for one of the experiments.

4.2.2.1. User sessions clustering results: AGENTS pages. Experiment Study: For this study, a small section (2556 log entries) of the web logs pertaining to the Agents homepage (http://www.csee.umbc.edu/agents/) were used. After running it through the clustering algorithm, we obtained 5 clusters.

4.2.2.2. Experimental observations

Cluster 1 (*Archives*) represents those users who are browsing the archive pages.
Clusters 2 and 3 have too small a cardinality to be included in the study.
Cluster 4 (*News*) represents users who are interested in the latest news (~33%) about 'Agents' technology.
Cluster 5 (*General Browser*) corresponds to users who usually navigate the main page (/agents/) and the links on that page. We can consider such users to be general browsers who are interested in knowing about the 'Agent' technology in general. The high probability of their navigating the introduction pages (~36%) and pages related to 'papers' (~30%) confirm our assumption.

5. Web query tool and preliminary result

We have created a web based thin client as the user interface to the system. By using a web browser the user interface becomes platform independent unlike proprietary tools such as Discoverer. The interface is exported as an HTML forms document from the web server that interacts with the Warehouse using CGI and SQLPLUS. The web interface was created using Perl CGI scripts. The user can enter (or select) a value into the fields, and the

Table 6. Subset of user session clusters for agents log.

Cluster no.	Cluster cardinality (no. of sessions)	No. of URLs in the cluster	URLs	Degree with which URL belongs to the cluster
1	19	56	{/agentslist/archive/digests}	0.21
Archives			or {/agentslist/archive/digests/}	
			{/agents/}	0.158
			{ /agentslist/archive/}	0.105
			Other URLs	<0.05
2	2	8	{/agents/commercial}	1.0
			{/agentslist/archive/1996..}	0.50
			{/agents/papers/migratory.html}	0.50
3	2	6	{/agents/}	1.0
			{/agents/groups}	0.5
			{/agents/commercial}	1.0
			{/agents/interface}	1.0
4	61	151	{/agents/} or {/agents}	0.72
News, FAQs,			{/agentslist/} or {/agentslist}	0.361
Technology			{/agents/news/}	0.213
			{/agents/technology/}	0.18
			{/agents/faq/}	0.164
			{/agents/agentnews/}	0.115
			Other URLs	<0.1
5	83	257	{/agents/}	0.783
General browser			{/agents/introduction/}	0.265
			{/agents/papers/}	0.169
			{/agents/web/}	0.132
			{/agents/papers/collections.shtml}	0.132
			{/agents/theory/}	0.12
			{/agents/mobile/}	0.12
			{/agents/faq/}	0.11
			{/agents/introduction/jennings98.pdf}	0.096
			{/agents/ standards/}	0.096
			{/agents/news/}	0.096
			Other URLs	<0.09

system automatically constructs an appropriate SQL query. The generated query is piped to SQLPLUS, which runs the query saved in the file and sends the result back to the interface. The result page shows the query that was used by the database along with the retrieved data. It also shows a count of the records retrieved. The web interface is very simple and user friendly. The users do not need to be aware of the underlying database architecture or SQL to use this tool optimally.

5.1. Ad-hoc analysis

The web interface allows users to perform ad-hoc analysis of both the web log warehouse and the mining results. By integrating both the components on a single tool, we have made it easier to analyze the access trends. A large number of analysis queries were given to the system and it produced correct results. We present some examples next.

Figures 4 and 5 show the web based query interface of the log analysis part of our system to the warehouse. The interface provides pull down menus and fill in fields that allow rollup, drill down and slicing queries. For example, for the Status dimension, users can query for all status values, or can drill down and query for log entries whose status values fall in HTTP response ranges such as 100s, 200s etc. The user can also query for individual status codes like 200 (success) or 404(error). This system can also handle HTTP 1.1 error codes and Methods. By default, the attributes have '%' in their fields, which is the wildcard character that aids in ad-hoc queries. E.g. If we want to query on the frequency of users accessing our web site from educational institutions, then we would enter '%.edu' on the Domain field of our interface. Or by entering '%Jun 1998' on the date field we can retrieve all records that were logged in June 1998. Figure 4 illustrates such an example. Thus arbitrary OLAP queries can be performed interactively by the user.

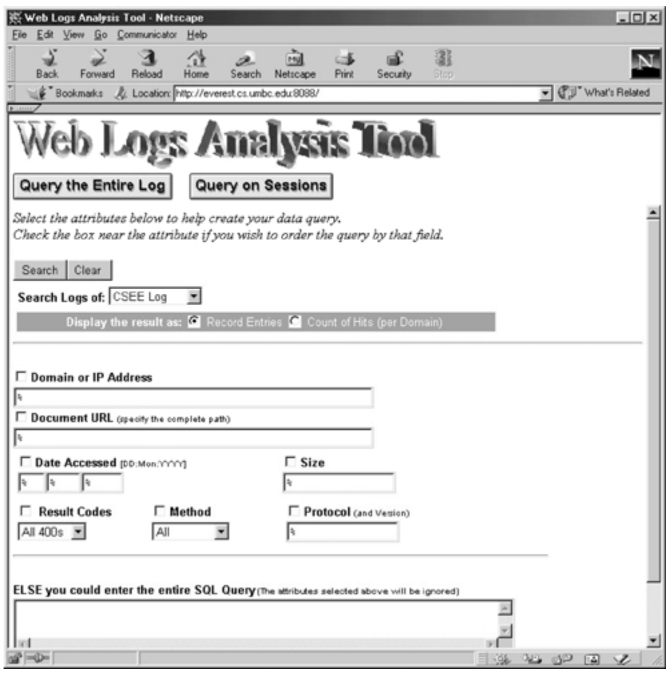
5.2. *Web interface for the warehouse*

Figures 4 and 5 show the web interface of our data warehouse. The interface provides pull down menus and fill in fields that allow rollup, drill down and slicing queries. For example, for the Status dimension, users can query for all status values, or can drill down and query for log entries whose status values fall in HTTP response ranges such as 100s, 200s etc. The user can also query for individual status codes like 200 (success) or 404(error). This system can detect HTTP 1.1 error codes and Methods. By default, most of the attributes have '%' in their fields. This is the wildcard character for Oracle database that aids in ad-hoc queries. E.g. If we want to query on the frequency of users accessing our web site from educational institutions, then we would enter '%.edu' on the Domain field of our interface. Or by entering '%Jun 1998' on the date field we can retrieve all records that were logged in June 1998. Figure 4 illustrates such example. Thus arbitrary OLAP like queries can be performed interactively by the user. Since the design of our database is hidden from the users, they can query the data as if they were trying to retrieve information from a transactional database. Currently, we query just one fact table in our database. We are enhancing our warehouse to query more fact tables that would aid users in generating more ad-hoc queries.

The Users who are aware of the database structure have additional flexibility of writing on-the-fly SQL queries to do arbitrary analyses (figure 5). This feature of our tool also allows dicing queries (e.g. 'select * from whole_log_fact1 where (doc_size>=200 and doc_size < 1000);').

The web interface is very simple and user friendly. The users do not need to be aware of the underlying database architecture or SQL to use this tool optimally.

Figures 6–8 show the web interface to our mining data. The interface provides pull down menus and fill in fields that allow users to query the clusters, sessions and the URLs for each application log. By default, most of the attributes have '%' in their fields. This is the wildcard character for Oracle database that aids in ad-hoc queries. E.g. If we want to query on the clusters of users accessing our web site from educational institutions, then we


Web Logs Analysis Tool - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://everest.cs.umbc.edu:8088/
What's Related
Web Logs Analysis Tool
Query the Entire Log
Query on Sessions
Search
Clear

Select the attributes below to help create your data query.
Check the box near the attribute if you wish to order the query by that field.

Search Logs of: CSEE Log

Display the result as: [✓ Record Entries] Count of Hits (per Domain)

☐ Domain or IP Address: %

☐ Document URL (specify the complete path): %

☐ Date Accessed [DD:Mon:YYYY]: % % %

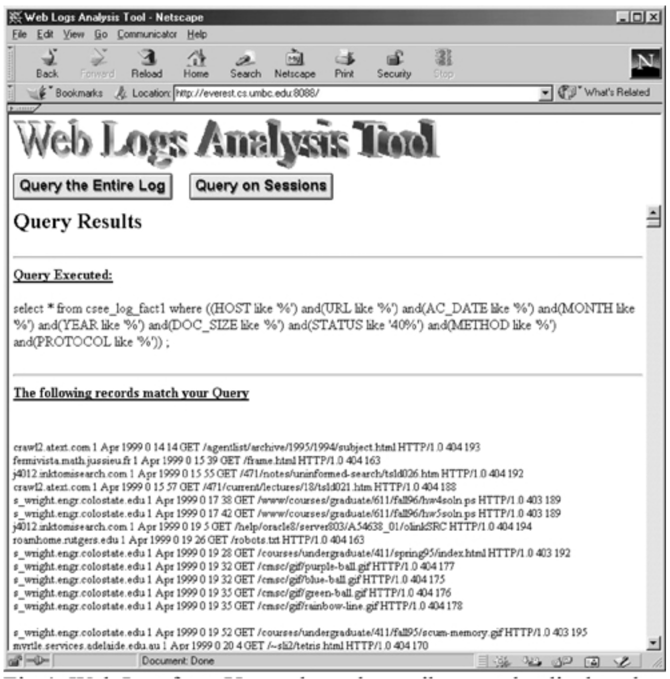
☐ Size: %

☐ Result Codes: All 400s

☐ Method: All

☐ Protocol (and Version): %

ELSE you could enter the entire SQL Query (The attributes selected above will be ignored)


Web Logs Analysis Tool - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://everest.cs.umbc.edu:8088/
What's Related
Web Logs Analysis Tool
Query the Entire Log
Query on Sessions
Document: Done

Query Results

Query Executed:

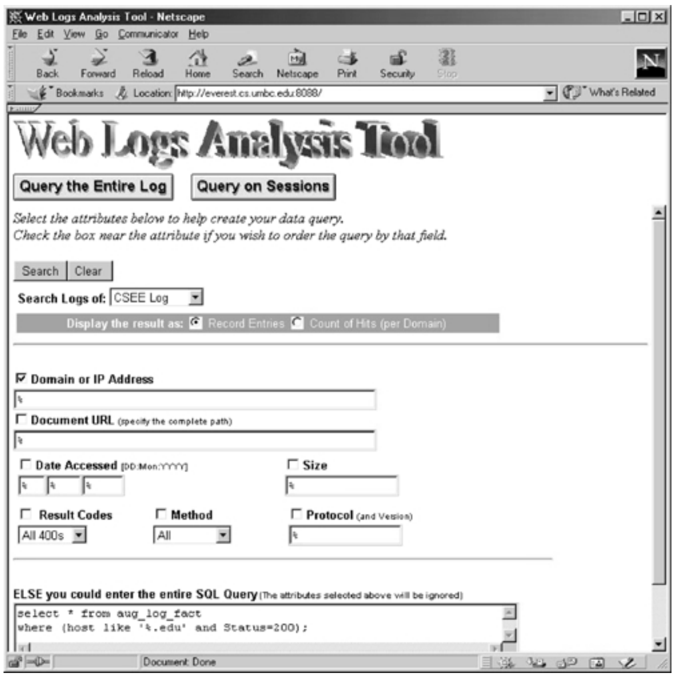
```
select * from csee_log_fact1 where ((HOST like '%') and(URL like '%') and(AC_DATE like '%') and(MONTH like
'%') and(YEAR like '%') and(DOC_SIZE like '%') and(STATUS like '40%') and(METHOD like '%')
and(PROTOCOL like '%')) ;
```

The following records match your Query

```
crawl2.atext.com 1 Apr 1999 0 14 14 GET /agentlist/archive/1995/1994/subject.html HTTP/1.0 404 193
femivista.math.jussieu.fr 1 Apr 1999 0 15 39 GET /frame.html HTTP/1.0 404 163
j4012.inktomisearch.com 1 Apr 1999 0 15 55 GET /471/notes/uninformed-search/tsld026.htm HTTP/1.0 404 192
crawl2.atext.com 1 Apr 1999 0 15 57 GET /471/current/lectures/18/tsld021.htm HTTP/1.0 404 188
s_wright.engr.colostate.edu 1 Apr 1999 0 17 38 GET /www/courses/graduate/611/fall96/hw4soln.ps HTTP/1.0 403 189
s_wright.engr.colostate.edu 1 Apr 1999 0 17 42 GET /www/courses/graduate/611/fall96/hw5soln.ps HTTP/1.0 403 189
j4012.inktomisearch.com 1 Apr 1999 0 19 5 GET /help/oracle8/server803/A54638_01/olinkSRC HTTP/1.0 404 194
roamhome.rutgers.edu 1 Apr 1999 0 19 26 GET /robots.txt HTTP/1.0 404 163
s_wright.engr.colostate.edu 1 Apr 1999 0 19 28 GET /courses/undergraduate/411/spring95/index.html HTTP/1.0 403 192
s_wright.engr.colostate.edu 1 Apr 1999 0 19 32 GET /cmsc/gif/purple-ball.gif HTTP/1.0 404 177
s_wright.engr.colostate.edu 1 Apr 1999 0 19 32 GET /cmsc/gif/blue-ball.gif HTTP/1.0 404 175
s_wright.engr.colostate.edu 1 Apr 1999 0 19 35 GET /cmsc/gif/green-ball.gif HTTP/1.0 404 176
s_wright.engr.colostate.edu 1 Apr 1999 0 19 35 GET /cmsc/gif/rainbow-line.gif HTTP/1.0 404 178

s_wright.engr.colostate.edu 1 Apr 1999 0 19 52 GET /courses/undergraduate/411/fall95/scum-memory.gif HTTP/1.0 403 195
myrtle.services.adelaide.edu.au 1 Apr 1999 0 20 4 GET /~sli2/tetris.html HTTP/1.0 404 170
```

Figure 4. Web interface: User selects the attributes to be displayed.


Web Logs Analysis Tool - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://everest.cs.umbc.edu:8088/
What's Related
Web Logs Analysis Tool
Query the Entire Log
Query on Sessions
Search
Clear
Document: Done

Select the attributes below to help create your data query.
Check the box near the attribute if you wish to order the query by that field.

Search Logs of: CSEE Log

Display the result as: [✓ Record Entries] Count of Hits (per Domain)

☑ **Domain or IP Address** %

☐ **Document URL** (specify the complete path) %

☐ **Date Accessed** [DD:Mon:YYYY] % % %

☐ **Size** %

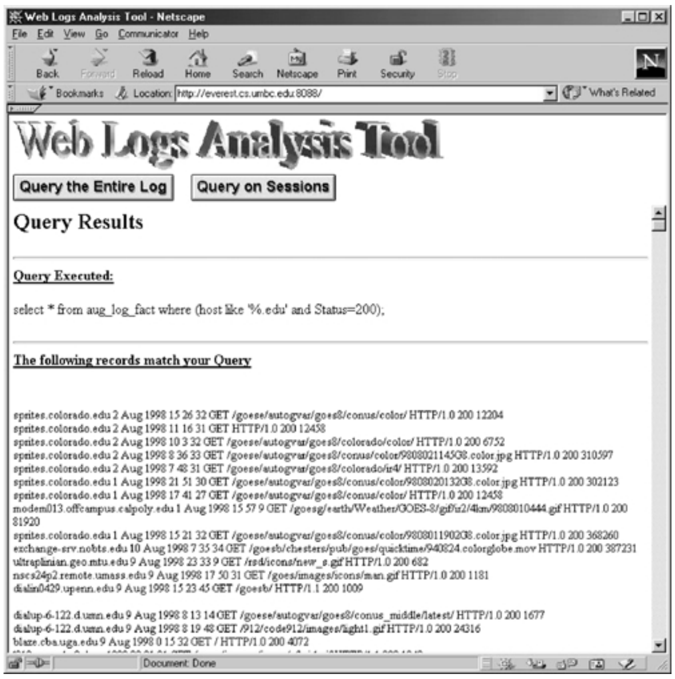
☐ **Result Codes** All 400s

☐ **Method** All

☐ **Protocol** (and Version) %

ELSE you could enter the entire SQL Query (The attributes selected above will be ignored)

```
select * from aug_log_fact
where (host like '%.edu' and Status=200);
```


Web Logs Analysis Tool - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://everest.cs.umbc.edu:8088/
What's Related
Web Logs Analysis Tool
Query the Entire Log
Query on Sessions
Document: Done

Query Results

Query Executed:

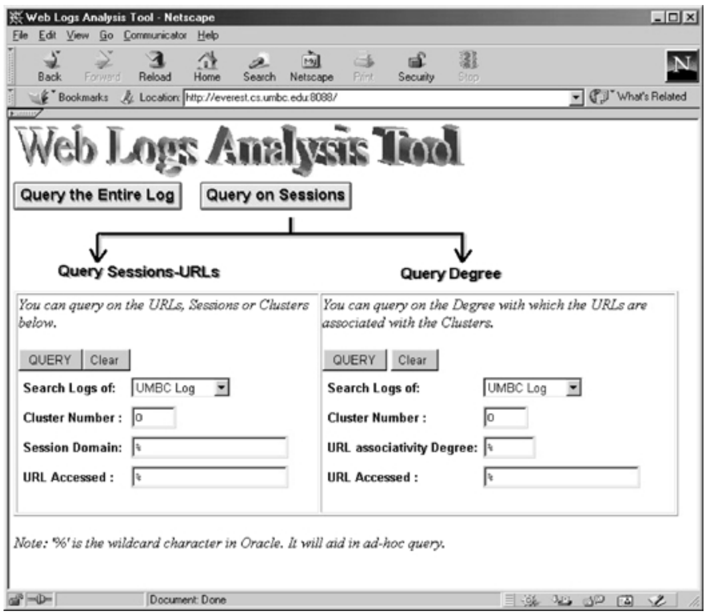
select * from aug_log_fact where (host like '%.edu' and Status=200);

The following records match your Query

```
sprites.colorado.edu 2 Aug 1998 15 26 32 GET /goese/autogvar/goes8/conus/color/ HTTP/1.0 200 12204
sprites.colorado.edu 2 Aug 1998 11 16 31 GET HTTP/1.0 200 12458
sprites.colorado.edu 2 Aug 1998 10 3 32 GET /goese/autogvar/goes8/colorado/color/ HTTP/1.0 200 6752
sprites.colorado.edu 2 Aug 1998 8 36 33 GET /goese/autogvar/goes8/conus/color/9808021145G8.color.jpg HTTP/1.0 200 310597
sprites.colorado.edu 2 Aug 1998 7 48 31 GET /goese/autogvar/goes8/colorado/ir4/ HTTP/1.0 200 13592
sprites.colorado.edu 1 Aug 1998 21 51 30 GET /goese/autogvar/goes8/conus/color/9808020132G8.color.jpg HTTP/1.0 200 302123
sprites.colorado.edu 1 Aug 1998 17 41 27 GET /goese/autogvar/goes8/conus/color/ HTTP/1.0 200 12458
modem013.offcampus.calpoly.edu 1 Aug 1998 15 57 9 GET /goesg/earth/Weather/GOES-8/gif/ir2/4km/9808010444.gif HTTP/1.0 200
81920
sprites.colorado.edu 1 Aug 1998 15 21 32 GET /goese/autogvar/goes8/conus/color/9808011902G8.color.jpg HTTP/1.0 200 368260
exchange-srv.nobts.edu 10 Aug 1998 7 35 34 GET /goesb/chesters/pub/goes/quicktime/940824.colorglobe.mov HTTP/1.0 200 387231
ultraplinian.geo.mtu.edu 9 Aug 1998 23 33 9 GET /rsd/icons/new_s.gif HTTP/1.0 200 682
nscs24p2.remote.umass.edu 9 Aug 1998 17 50 31 GET /goes/images/icons/man.gif HTTP/1.0 200 1181
dialin0429.upenn.edu 9 Aug 1998 15 23 45 GET /goesb/ HTTP/1.1 200 1009

dialup-6-122.d.umn.edu 9 Aug 1998 8 13 14 GET /goese/autogvar/goes8/conus_middle/latest/ HTTP/1.0 200 1677
dialup-6-122.d.umn.edu 9 Aug 1998 8 19 48 GET /912/code912/images/light1.gif HTTP/1.0 200 24316
blaze.cba.uga.edu 9 Aug 1998 0 15 32 GET / HTTP/1.0 200 4072
```

Figure 5. Web interface: User enters an SQL query. The tool ignores the entries in the attribute.


Web Logs Analysis Tool - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://everest.cs.umbc.edu:8088/ What's Related
Web Logs Analysis Tool
Query the Entire Log
Query on Sessions
Query Sessions-URLs
Query Degree
QUERY
Clear
Document: Done

You can query on the URLs, Sessions or Clusters below.

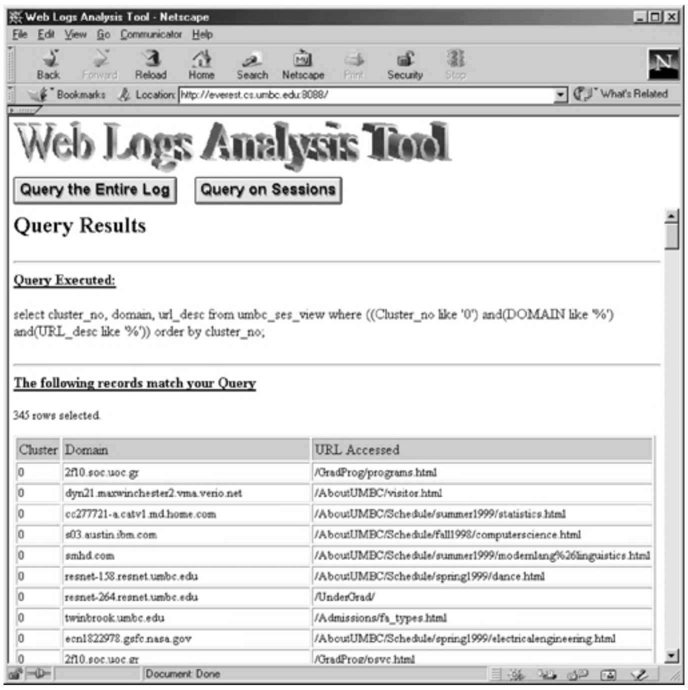
Search Logs of:	UMBC Log
Cluster Number :	0
Session Domain:	%
URL Accessed :	%

You can query on the Degree with which the URLs are associated with the Clusters.

Search Logs of:	UMBC Log
Cluster Number :	0
URL associativity Degree:	%
URL Accessed :	%

Note: '%' is the wildcard character in Oracle. It will aid in ad-hoc query.

Figure 6. Interface to the data mining component.


Web Logs Analysis Tool - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://everest.cs.umbc.edu:8088/ What's Related
Web Logs Analysis Tool
Query the Entire Log
Query on Sessions
Document: Done

Query Results

Query Executed:

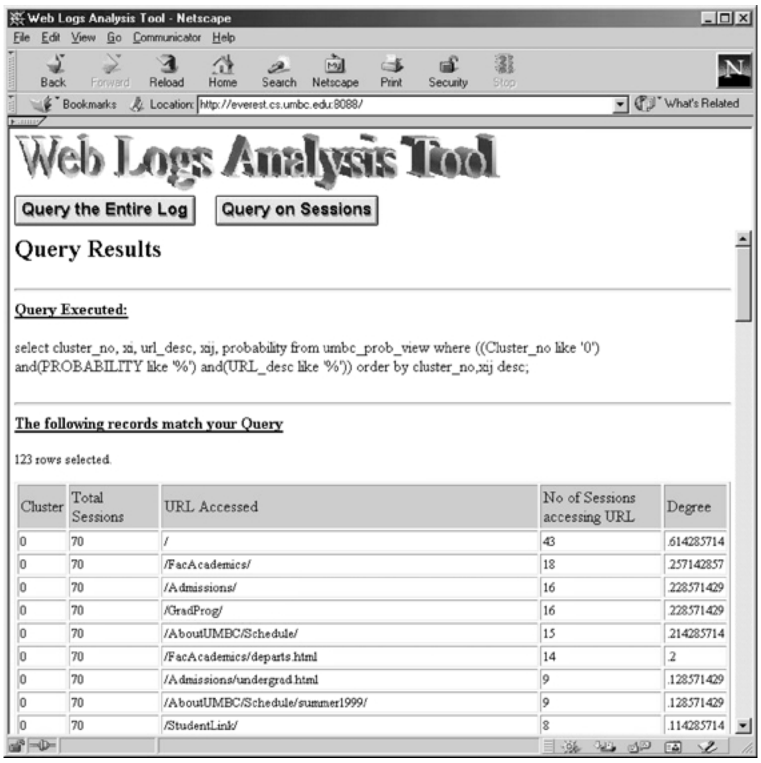
```
select cluster_no, domain, url_desc from umbc_ses_view where ((Cluster_no like '0') and(DOMAIN like '%')
and(URL_desc like '%')) order by cluster_no;
```

The following records match your Query

345 rows selected.

Cluster	Domain	URL Accessed
0	2f10.soc.uoc.gr	/GradProg/programs.html
0	dyn21.maxwinchester2.vma.verio.net	/AboutUMBC/visitor.html
0	cc277721-a.catv1.md.home.com	/AboutUMBC/Schedule/summer1999/statistics.html
0	s03.austin.ibm.com	/AboutUMBC/Schedule/fall1998/computerscience.html
0	smhd.com	/AboutUMBC/Schedule/summer1999/modernlang%26linguistics.html
0	resnet-158.resnet.umbc.edu	/AboutUMBC/Schedule/spring1999/dance.html
0	resnet-264.resnet.umbc.edu	/UnderGrad/
0	twinbrook.umbc.edu	/Admissions/fa_types.html
0	ecn1822978.gsfc.nasa.gov	/AboutUMBC/Schedule/spring1999/electricalengineering.html
0	2f10.soc.uoc.gr	/GradProg/psyc.html

Figure 7. Query output for the sessions-URL part.


Web Logs Analysis Tool - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://everest.cs.umbc.edu:8088/
What's Related
Web Logs Analysis Tool
Query the Entire Log
Query on Sessions

Query Results

Query Executed:

```
select cluster_no, xi, url_desc, xij, probability from umbc_prob_view where ((Cluster_no like '0')
and(PROBABILITY like '%') and(URL_desc like '%')) order by cluster_no,xij desc;
```

The following records match your Query

123 rows selected.

Cluster	Total Sessions	URL Accessed	No of Sessions accessing URL	Degree
0	70	/	43	.614285714
0	70	/FacAcademics/	18	.257142857
0	70	/Admissions/	16	.228571429
0	70	/GradProg/	16	.228571429
0	70	/AboutUMBC/Schedule/	15	.214285714
0	70	/FacAcademics/departs.html	14	.2
0	70	/Admissions/undergrad.html	9	.128571429
0	70	/AboutUMBC/Schedule/summer1999/	9	.128571429
0	70	/StudentLink/	8	.114285714

Figure 8. Query output for the cluster-URL part.

would enter '%.edu' on the Domain field of our interface. Or by entering '0' on the Cluster Number field we can retrieve all URLs that belong to the sessions of that cluster. Figure 8 illustrates such example. Thus arbitrary OLAP like queries can be performed interactively by the user.

6. Future enhancements

Currently, we have populated the fact table directly from the web logs. We can augment our system by loading the data into the transactional database, and then populate the fact tables from it. Stored procedures or triggers could be written to populate specific fact tables with web log as they get added, or in a batch mode. By creating such a transactional database, the pre-processing step will be automated to a large extent, and more dynamic "monitoring" can be done by the system automatically. Features like alerts and warnings can be easily incorporated in this architecture. For example, access patterns that denote security violations would be identified as they happen by the system (e.g. the abuse of the CGI phf script, or attempted buffer overflow attacks). If a security violation is taking place, the site administrator can be alerted or the web site could be made inaccessible. This proactive feature of our architecture can aid in more efficient web site administration. However this requires modifying the logging module in the web server to store access logs directly into the database.

Finally, our experimentation with the web mining was done with single (most reasonable) values for many of the tuneable parameters, such as the time delta involved in sessionizing logs, confidence and support for associations, initializing of the medoids in clustering etc. Clearly, it would be interesting to experiment with and see the effect of these parameters on the eventual results.

Acknowledgments

This work was partially supported by NSF award (IIS 9801711) to Anupam Joshi. We would also like to thank the anonymous referees for their comments which have helped us improve this paper.

References

1. 'A Listing of Access Log Analyzers', http://www.uu.se/Software/Analyzers/Access-analyzers.html.
2. R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, Sept. 1994.
3. H. Ahonen, O. Heinonen, M. Klemettinen, and I. Verkamo, "Mining in the phrasal frontier," in 1st European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'97), Norway, June 1997.
4. A. Buchner and M. Mulvenna, "Discovering internet marketing intelligence through online analytical web usage mining," SIGMOD Record, vol. 27, no. 4, pp. 54–61, 1998.
5. M.S. Chen, J.-S. Park, and P.S. Yu, "Efficient data mining for path traversal patterns," IEEE Trans. on Knowledge and Data Engineering, vol. 10, no. 2, pp. 209–221, 1998.
6. R. Cooley, B. Mobasher, and J. Srivastava, "Web mining: Information and pattern discovery on the world wide web," in ICTAI'97, Dec. 1997, pp. 558–567.
7. 'Follow: A session based Log analyzing tool,' http://www.pobox.com/∼mnot/follow/.
8. A. Joshi and R. Krishnapuram, "Robust fuzzy clustering methods to support web mining," in Proc. Workshop in Data Mining and knowledge Discovery, SIGMOD 1998.
9. A. Joshi and R. Krishnapuram, "On mining web acceess logs," in Proc. SIGMOD 2000 Workshop on Research Issues in Data Mining and Knowledge Discovery, Dallas, 2000, pp 63–69.
10. T. Kamdar, MS Thesis, CSEE Department, University of Maryland Baltimore County, May 2001.
11. B. Lent, R. Agrawal, and R. Srikant, "Discovering trends in text databases," in Proc. of the 3rd Int'l Conference on Knowledge Discovery in Databases and Data Mining, Newport Beach, California, August 1997.
12. O. Nasraoui, H. Frigui, A. Joshi, and R. Krishnapuram, "Extracting web user profiles using relational competitive fuzzy clustering," Intl. J. Artificial Intelligence Tools, vol. 9, no. 4, pp. 509–526, 2000.
13. O. Nasraoui, R. Krishnapuram, and A. Joshi, "Mining web access logs using a fuzzy relational clustering algorithm based on a robust estimator," (poster) at WWW8, August 1999.
14. M. Perkowitz and O. Etzioni, "Towards adaptive web sites: Conceptual framework and case study," in Proc. of the Eighth International WWW Conference, May 1999, pp. 1245–1258.
15. SGI-MineSet 'http://www.sgi.com/software/mineset/'.
16. C. Shahabi, A.M. Zarkesh, J. Abidi, and V. Shah, "Knowledge discovery from user's web-page navigation," in Proc. Seventh IEEE Intl. Workshop on Research Issues in Data Engineering (RIDE), '97, pp. 20–29.

17. “SpeedTracer: A web usage mining and analysis tool,” IBM Systems Journal, vol 37, no. 1—Internet Computing, pp. 89–105, 1998.
18. L. Yi, R. Krishnapuram, and A. Joshi, “A fuzzy relative of the *k*-medoids algorithm with application to document and snippet clustering,” IEEE Int’l Conference—Fuzzy Systems, 1999.
19. O.R. Zaiane, M. Xin, and J. Han, “Discovering web access patterns and trends by applying OLAP and data mining technology on web logs,” in Proc. Advances in Digital Libraries Conf. (ADL’98), Santa Barbara, CA, April 1998, pp. 19–29.
20. A. Zarkesh, J. Adibi, C. Shahabi, R. Sadri, and V. Shah, “Analysis and design of server informative WWW-sites,” in Proceedings of the ACM CIKM’97, pp. 254–261.