Discovery of Web Frequent Patterns and User Characteristics from Web Access Logs: A Framework for Dynamic Web Personalization

Sumeet Dua,
Louisiana State University,
sumeetd@bit.csc.lsu.edu

Eungchun Cho
Kentucky State University,
eccho@gwmail.kysu.edu

S.S. Iyengar
Louisiana State University,
iyengar@bit.csc.lsu.edu

Abstract

An automatic discovery method that discovers frequent access routines for unique clients from web access log files is presented. Proposed algorithm develops novel techniques to extract the sets of all predictive access sequences from semi-structured web access logs. Important user access patterns are manifested through the frequent traversal paths, thus helping understand user surfing behaviors. The predictive access routines discovered by AllFreSeq are also useful for understanding and improving web-site domain tree.

1. Introduction:

Data Mining, commonly known as Knowledge discovery in databases has opened new avenues in the areas of database research. The purpose of data mining is to facilitate understanding large amounts of data by discovering interesting regularities and rules. The World–Wide-Web offers tremendous opportunities for marketers to reach a vast variety of audiences at a less cost than any other medium. Recent studies have shown that Web consumes more Internet bandwidth than any other application [11]. With huge amount of capital invested in these sites, it has become necessary to understand the effectiveness and realize the potential opportunities offered by these services.

Access log files provides a wealth of information about visitors to the web site. Popular web sites have their web logs growing at a tremendous speed adding hundreds of megabytes per day. Condensing these colossal files of raw web log data in order to retrieve significant and useful information is a non-trivial task. Current web log analysis programs are unable to accurately determine unique visits, users, and user demographics, or even access totals for each page after caching are taken into account. This research is a step closer to perform dynamic analysis on such huge amounts of data and help institutions make effective use of the web access history for server performance enhancement, system design improvement, customer targeting in Marketing industry and means to cross-tabulate data to see which web-items are been selected by certain audience segments. The primary important task is to uniquely identify the number of distinct users visiting your site. We call these distinct sessions as “unique threads”. After identifying these unique sessions from the access log we present techniques to determine for each of such unique users:

Number of accesses to the Web Server Number of times a Intermediate index pages are accesses from the home page Order of the page accesses by each of the unique users. This helps to evaluate the train of thought of the users Frequent access patterns for each of these Unique users Cyclic links within the Client/Web Server domain

This knowledge of common and popular patterns can help us in the design of static hypertext organization that requires user intervention. More interestingly, we present a framework to customize the web site on the fly and dynamically link hypertext pages for individual users. The objective is to match an active user’s access pattern with most of the categories discovered from our algorithm. Beside by using dynamic web server suggestion, we may put user category information to improve server performance. The server may pre-fetch pages that a user is likely to visit soon, based on what he has accesses and what category he falls into. Needles to mention combination of these two frameworks can be used to design a Dynamic Web server with high performance index and efficient information retrieval.

1.1 Web Access log model:

Web server access log files contains valuable information about the user access behavior. A web server access file entry contains the following entries:

- Host client Internet Protocol (IP) address
- Time stamp
- Method employed
- URL (Uniform Resource Locator) address of the accesses document
- HTTP (Hyper Text Transfer Protocol) version
- Return code (status of the request)

0-7695-0559-7/00 $10.00 © 2000 IEEE

- Bytes transferred
- Referrer page URL
- Browser used
- Client Operating System

205.188.209.71 [26/Nov/1999:19:35:34 -0600] "GET/home.html HTTP/1.0" 20 0 401 “http://www.lsu.edu” “Mozilla/4.0”
205.188.209.71 [26/Nov/1999:19:35:37 -0600] "GET/research/research.html HTTP/1.0" 20 0 401 “/index.html” “Mozilla/4.0” 205.188.209.71 [26/Nov/1999:19:35:49 -0600] "GET/research/lab/index.html HTTP/1.0" 20 0 401 “/research/research.html” “Mozilla/4.0”

Figure 1: Sample Web Access Log in Extended Log Format

A sample set of entries in a web access log is shown is figure 1. Unique user identifier is usually not available in the log file. The closet estimate is the number of “unique sessions” which is determined by the number of different IP addresses in the log file. However this method of estimating these unique sessions is less than accurate. This is because:

1. Dial-up users are usually assigned a different IP address each time they connect to the ISP
2. Users of commercial online services (e.g. AOL, MSN) access the internet through a number of Internet “gateways” each of which has their IP address logged in the Access log
3. Users from the organization intranet connect through proxy computers each with a IP address (that is logged)
4. Public terminals, e.g. in libraries, allow many users to use the same computer.

This implies that we need a Profile tracking system to uniquely identify a unique session. Next section describes a technique to uniquely identify a user session.

1.2 Organization of the paper:

The rest of the paper is organized as follows. Section 2 describes the Discovery process of Unique Client access routines. Section 3 simulates the proposed approach with experimental data. Section 4 proposes some applications of the approach. We conclude with a summary in section 5. Some future work is proposed in section 6 and section 7 lists the references.

2. Discovery of Unique Client’s Access Routines

We cast a problem of identification of users unique access sequences. These sequences are used as a input to our algorithms to discover frequent access routines for each client that satisfy the condition of minimum support.

2.1 Log Thread

Before we proceed further, let us define a parameter called log thread. A log thread is a non-empty unordered collection of items (and without loss of generality, we assume that all access pairs of a log thread are sorted in increasing order). For each log record, we use a referral page and the accessed page to form a hyper linked log thread. In addition to this also we also associate accessed URLs of the neighboring log entries. More formally,

If (R_i, U_i) represents a access pair , then a log thread of a session S can be expressed as:

$TLS = \{(R_1, Y_1), (R_2, Y_2), \ldots, (R_n, Y_n)\}$

Where, $R_{i+1} = Y_i$, $1 \leq i < n$.

A new pair (Rj, Yj) can be appended to the thread if $Rj = Yk$, $1 \leq k \leq n$, or $R_1 = Y_j$

Now, the local clients or various proxy servers generally cache some web pages, or both, to reduce network traffic. So the entries for such accesses to the logged pages, result in an inconsistency. These break points in the threads can be accounted for by building the backward paths using the domain knowledge of the Web site. A sample domain knowledge tree of a web site is shown in Appendix 1.

So the rules for appending a new access pair (Rj, Yj) to the web log demand the following heuristics:

1. A new pair (Rj, Yj) can be appended to the thread if $Rj = Y_k$, $1 \leq k \leq n$, or $R_1 = Y_j$
2. If $Rj \neq Y_n$, a backward access path , derived the web domain knowledge tree, must be first added before the access pair is appended. This ensures that there is a legitimate flow of data paths of accesses URLs in the identification of Log thread. There can be multiple candidate threads to which the access pair can be appended. For simplicity, we use a rule that the access pair must be appended to the log thread if and only if the number of access logs required for building the breakpoints is least among candidate threads. This ensures that same sub-threads do not appear in different threads.

After we have appended sufficient access pairs to the log thread table, the log thread can be presented as the

sequence of accesses URLs. Thus a log thread can now be represented as:

$TLS = \{R_1, R_2, \ldots, R_n\}$
TLS=(/home.html,
/research/research.html, /research/lab/index.html,
/research/lab/cclms/cclms.html,
/research/lab/cclms/projects/index.html,
/research/lab/cclms/projects/parallel/index.html)

Figure 2: Log Thread

2.2 Identification of unique session:

We use the following attributes of the access log file entry to uniquely identify a user session:
Cid_i = IP address of the Client
T_i = Timestamp of access
U_i = URL accessed by the client
R_i = Referral page
Different Cid indicates different user sessions. If Cid_i (Cid_j then the access belong to different user sessions. If the condition $Cid_i = Cid_j$ is satisfied and $T_j - T_i < T_{threshold}$, where $T_{threshold}$ is the predetermined time width of access, then the access are for unique user sessions If this conditions is not satisfied then the accesses are for different user sessions.

2.3 Identification of Passage Sequence:

Once we have identified unique sessions and the corresponding log entries, we build a log thread structure for each of these sessions. If the above criterions for unique sessions are satisfied, then a log thread is identified for the entries that have same Cid and $T_j - T_i$ ($T_{threshold}$.
The Log thread, TL, has a unique identifier and contains a set of access pairs. A Client, C, has a unique identifier and has a associated list of Passage sequence P which is a triplet of (T_{start} , T_{end} , TL) where:
Tstart is the starting time of the Passage sequence and Tend is the ending time of the Passage sequence. TL is an ordered log thread of the unique client

P205.188.209.71={26/Nov/1998:19:35:34 –0600,11/Dec/1998:14:21:35–0600,
[(/home.html,/research/research.html,
/research/lab/index.html, /research/lab/rrl/index.html,
/research/lab/rrl/mission.html),
(/home.html,/research/research.html,
/research/lab/index.html,
/research/lab/cclms/cclms.html,
/research/lab/cclms/projects/index.html,
/research/lab/cclms/projects/parallel/index.html,/research/lab/cclms/projects /parallel/index.html),

Figure 3: Passage Sequence for Client 205.188.209.71(26/Nov/1998:19:35:34 –0600, 11/Dec/1998:14:21:35 –0600)

Without loss of generality we assume that, no Client has more than one Passage Sequence with the same time stamp, so that we can use (Tstart, Tend) as the Sequence identifier. We also assume, as indicated earlier, that the list of Client's Passage Sequences are sorted in time. Thus the list of Client's Passage sequence is itself a sequence (P1, P2… Pn). A sample passage sequence is shown in figure 3.
A Client –sequence, C, is said to contain a sequence ((C, that is, (is a subsequence of the Client-sequence C. The support or frequency of a sequence (is the fraction of the total Passage Sequences that contain the query sequence (, i.e.
Given a application defined threshold called the minimum support (denoted minimum_sup), we say that a sequence is frequent if fr((, D) (minimum_sup.
Now the problem of mining web access patterns can be formulated as follows:
Given a Passage Sequence P of Client Log threads, the problem of mining access patterns is to find all frequent Log sub-threads in the Passage Sequence.

2.4 Access Tree:

After we have identified unique passage sequences for each of the unique clients we mark the occurrences of access routine in a sparse access graph G (V, E), where |V| is the number of vertices and |E| is the number of edges. Since in such a graph there is one and only one path between every pair of vertices, such graph is an access tree from the following theorem:
Theorem 1: If in a graph there is one and only one path between every pair of vertices, G is a tree.
Proof: Existence of a path between every pair of vertices assures that G is a connect graph. A circular link (circuit) in a graph with more than two vertices implies that there is at least one pair of vertices a, b such that there is two distinct path between a and b. Since G has only one path between every pair of vertices, G can have no circuits. Therefore G is a tree.
Each leaf of Access tree is a data structure of the type Node that is a tuple of Parent Node(s), Weight of the node, Child Node(s). The exact data structure is described in figure 4:

```
Type node
{
Parent [I], I ( [1..N]
Weight
Child [J], J ( [1..M]
}
```

Figure 4: Node data structure

node.parent[I] contains the name of the parent node of whose current node is a child. This attribute is empty when the node is first formed. node.weight attribute contains the number of times the node appears in the sequence of a client access. The value is initialized to one when the node is first formed and is incremented as the node appears again. node.child[j] contains the name of the child node for which the current node is a parent. This attribute is empty for the last node in the sequence.

To describe the running time of traversing such a tree we represent it in the form of G=(V, E) where input is in terms of number of vertices |V| and number of edges |E| of the graph.

2.5 Knowledge Discovery Process:

The next step is to discover frequent branches that have a minimum support of minimum_supp. For experimental purposes we assume a minimum support = 25%. This means that the given branch appears at least in 25% of the client's log threads. We present an algorithm that discovers all sequences and their absolute support value. As the next step we identify the sequences that satisfy the minimum support criterion. The algorithm AllFreqSeq has two main steps. As the first step we identify the end node using Depth first search technique and then as a second step we backtrack to the root node and calculate the absolute support value. The algorithm is formally described in figure 6:

Let G be the given Client's access tree. Let R be the root vertex from where the search begins. Let FREQ and FREQPASS be the subset into which G has to partition

Algorithm AllFreqSeq:
Input: Clients Access Tree G(V, E), minimum_supp
Output: FREQPASS -> Frequent sequences with support value
Main algorithm Start
Step 1: Set v=R, I=0, FREQ = NULL, FREQPASS = NULL
Step 2: Set I=I+1, support(FREQ) = 0
Step 3: Look for a un-transversed edge incident on v If there is no such edge (i.e., every edge incident on v has already been transversed) go to step 5. Otherwise:
Pick the first untraversed edge at v, say (v, w) and traverse this edge. Orient the edge (v, w) from v to w. Now you are at vertex w.
Step 4: Add edge (v, w) to the set FREQ. Set v=w and go to step 2.
Step 5: If w.weight in FREQ < minimum_supp then
Delete edge (v, w) from FREQ
If no then Support(FREQ) = w.weight
Delete edge (v, w) from G
Step 6: Check if there exists some traversed edge (u, v) in set FREQ oriented towards v
If there is such an edge move back to vertex u (Note that u is the vertex from which v was visited for the first time).
Weight(v) = weight(v) – support(FREQ)
If weight(v) = 0 then delete (u, v) from G
Set v=u
Goto Step 5
If there is no such edge (u, v), stop (we are back at root x, having traversed one FREQ sequence). Check if there is an untraversed edge on u:
If yes then
If FREQ <> NULL, Add FREQ to FREQPASS and set FREQ = NULL
Set v=u and goto step 3
Else goto step 7
Step 7: Exit
Main algorithm End

Figure 6: Algorithm AllFreqSeq:

3. Experimental results:

Let us consider the log entries shown in figure 7 of the access log file. The Access log file is in Extended log format structure.

```
205.188.209.71 [26/Nov/1998:19:35:34 -0600] "GET/home.html HTTP/1.0" 20 0 401 “http://www.lsu.edu” “Mozilla/4.0” 205.188.209.71 [26/Nov/1998:19:35:37 -0600]"GET/research/research.html HTTP/1.0"200401“/home.html” “Mozilla/4.0” 205.188.209.71 [26/Nov/1998:19:35:49 -0600]"GET/research/lab/index.html HTTP/1.0"200401 “/research/research.html” “Mozilla/4.0”
```

Figure 7: Access log entries in extended log format

Log threads are formed from the entries in figure 7 using the techniques described in section 2.1 The list of log threads extracted from the above log entries is given in figure 8.

TL1=(/home.html, /research/research.html, /research/lab/index.html, research/lab/rrl/index.html, /research/lab/rrl/mission.html)
TL2=(/home.html, /research/research.html, /research/lab/index.html, /research/lab/cclms/cclms.html, /research/lab/cclms/projects/index.html, /research/lab/cclms/projects/parallel/index.html,/research/lab/cclms/projects /parallel/index.html)
TL3=(/home.html,/research/research.html, /research/groups/groups.html)
TL4 = (/home.html, /faculty/faculty.html)

Figure 8: Log threads for Access log entries

The corresponding Passage Sequence for the above log threads is shown in figure 3. Using the techniques described in section 2.4 we form the access tree for the Passage sequence.

FrequentPassageSequence	SupportVal
W1 -> W2 -> W6	3
W1 -> W2 -> W7	3

Figure 9: Frequent Client Sequences

This graph now serves as an input to the algorithm AllFreqSeq described in figure 6. The result is the frequent episodes listed in figure 9 with their corresponding support values.

4. Applications:

Number of approaches has been developed dealing with specific aspects of Web usage mining for the purpose of automatically discovering user profiles. For example, Perkowitz and Etzioni [PE98] proposed the idea of optimizing the structure of Web sites based co-occurrence patterns of pages within usage data for the site. Schechter et al [SKS98] have developed techniques for using path profiles of users to predict future HTTP requests, which can be used for network and proxy caching. Spiliopoulou et al [SF99], Cooley et al [CMS99], and Buchner and Mulvenna [BM99] have applied data mining techniques to extract usage patterns from Web logs, for the purpose of deriving marketing intelligence. Shahabi et al [SZA97], Yan et al [YJGD96], and Nasraoui et al [NFJK99] have proposed clustering of user sessions to predict future user behavior.

Since our results show the access routines of a user over a desired period of time and with recommended support, the applications can be multi-fold. We propose the following architecture for the Dynamic web server and updating the web pages on-the fly as the client leads to surf the site further. The proposed architecture is shown in figure 10. This architecture is an approach to usage-based web personalization taking into account the full spectrum of our web access routine techniques. Updating web pages on the fly has promising avenues in e-commerce and web medical services sector.

The overall process of usage-based web personalization can be divided into two basic components. The offline component, involves accessing the web server log and predicting user access pattern. Once this task is successfully accomplished, the online component is now capable of providing positive recommendations to the clients. However this process can improve the system performance by almost a factor of 10. The mine engine can prefetch the pages, which the user is most likely to visit and even cache suitable pages.

5. Conclusion:

We have presented an automatic discovery method that discovers frequent access routines for unique clients. Proposed algorithm AllFreqSeq develops novel techniques to extract the sets of all predictive access sequences from semi-structured web access logs. Important user access patterns are manifested through the frequent traversal paths, thus helping understand user surfing behavior. The predictive access routines discovered by AllFreSeq were also extremely useful for understanding and improving web-site domain tree.

6. Future Work

This work opens several research opportunities, which we plan to address in the future. These include but are not limited to:

1. Real time application of AllFreqSeq on top a RDBMS containing dynamically updated web log entries.
2. Discovering general behavior patterns for all users.
3. Imposing a taxonomy on the behavioral pattern within a sliding window of time

7. References

[AGAR93] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases, " Proc. ACM SIGMOD Conf. Management of Data, pp. 207-216, Washington, D.C., May1993.

[BM99] Buchner, A. and Mulvenna, M. D., Discovering internet marketing intelligence through online analytical Web usage mining. SIGMOD Record, (4) 27, 1999.

[NARS95] Narsingh Deo, Graph theory with applications to engineering and Computer Science, Prentice Hall, 1995

[NFJK99] Nasraoui, O., Frigui, H., Joshi, A., Krishnapuram, R., Mining Web access logs using relational competitive fuzzy clustering. To appear in the Proceedings of the Eight International Fuzzy Systems Association World Congress, August 1999.

[PE98] Perkowitz, M. and Etzioni, O., Adaptive Web sites: automaticlly synthesizing Web pages. In Proceedings of Fifteenth National Conference on Artificial Intelligence, Madison, WI, 1998.

[SF99] Spiliopoulou, M. and Faulstich, L. C., WUM: A Web Utilization Miner. In Proceedings of EDBT Workshop WebDB98, Valencia, Spain, LNCS 1590, Springer Verlag, 1999.

[SKS98] Schechter, S., Krishnan, M., and Smith, M. D., Using path profiles to predict HTTP requests. In Proceedings of 7th International World Wide Web Conference, Brisbane, Australia, 1998.

[SZAS97] Shahabi, C., Zarkesh, A. M., Adibi, J., and Shah, V., Knowledge discovery from users Web-page navigation. In Proceedings of Workshop on Research Issues in Data Engineering, Birmingham, England, 1997.

[ZAKI1998] Zaki, Lesh, Ogihara, “PlanMine: Sequence Mining for Plan failures”