ISSN No. 0976-5697

Volume 2, No. 2, Mar-Apr 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Implementation of Web usage Mining tool using Efficient Algorithm

Kapil Sharma*
Information Technology deptt,
LNCT
Bhopal, India
Kapil.rjit@gmail.com

Prof. Vineet Richhariya
HOD CSE deptt,
LNCT
Bhopal, India
Vineet_rich@yahoo.co.in

Yogendra S. Rathore
Computer Science & Engg. deptt,
ITM
Gwalior, India
Yogendra.cse.2006 @gmail.com

Abstract: In this paper,The World Wide Web (WWW) continues to grow at an astounding rate in both the sheer volume of traffic and the size and complexity of Web sites. An important input to website design is the analysis of how a Web site is being used by the users. Usage analysis includes straightforward statistics, such as page access frequency, as well as sophisticated forms of analysis, such as finding the common traversal paths through a Web site. *Web Usage Mining* is usually done using data mining techniques to determine frequent access patterns of the users. There are two phases in the Web Usage Mining. The first phase is Data Preprocessing. There are several preprocessing tasks i.e. Data Cleaning, User Identification, Session Identification and Transaction Identification. These tasks are computationally intensive and time consuming. Which need to be performed on the data, this data collected from web server logs. The work initially develops on usage data preparation techniques in order to identify users and the user sessions. There after groups the user sessions of the users based on semantically meaningful access paths of the users. These access paths help to generate the frequent access patterns of the user that can be used for effective web page design. The second phase is Mining the frequent patterns. This phase generate the frequent patterns from the access paths of the users.

Keywords: Weblog, Access Paths, Patterns,mining,,min support

I. INTRODAUCTION

A. Web Mining

The Web contains huge amount of web sites. A *web site* usually contains great amounts of information distributed through hundreds of pages. Without proper guidance, a visitor often wanders aimlessly without visiting important pages, loses interest and leaves the site sooner than expected. This consideration is at the basis of the great interest about Web Usage Mining both in the academic and the industrial world. In the online shopping, E-commerce site if the user is not getting his required pages, he will simply switches to the another web site. For avoiding this we have to find the frequent user access patterns from his previous or history. Web Usage mining will generate user access patterns from the web log records. This Web log records are stored in the web server application server.

Web Usage Mining is one type of Web mining. architecture divides the Web usage mining process into two main parts. Input to the Web usage Mining process is Web Server Logs, Error logs, User Cookies and Client Cache Records. The first part includes the domain dependent processes of transforming the Web data into suitable transaction form. This includes preprocessing, transaction identification, and data integration components. The second part includes the largely domain independent application of generic data mining and pattern matching techniques (such as the discovery of association rule and sequential patterns) as part of the system's data mining engine. The below figure shows architecture of Web Usage Mining tool.

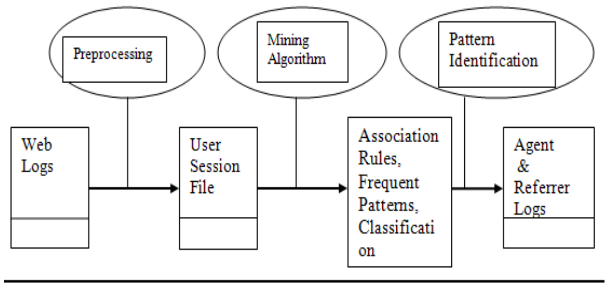

Figure 1.1. Web Usage Mining Architecture

B. Data Preprocessing

The input to the preprocessing phase is web logs. This file is stored in a common log format. This format [11] is given by W3C (World Wide Web Community). But which is in unstructured format for Web Usage Mining. The below figure shows the common log file data format.

```
<ClientIP><UserID><AccessTime><HTTPrequest><URL><Protocol><StatusCode
> <Agent>
```

Figure.2 Common Log Format in the Web Server

Client IP: Client IP address or hostname (if DNS lookups are performed)
UserID: User name ('-'if anonymous)
Access Time: dd/mm/yy, hh:mm:sec (date format)
HTTP request: HTTP methods are GET, POST, and HEAD…
Protocol: Protocol used for transmission (HTTP/1.0, HTTP/1.1)
Status Code: The status code returned by the server as 9response (200 o.k., 404 for not found….);
Bytes Transferred: The no. of bytes transferred for that particular request.
User Agent: Which identify the client application used to retrieve the resource?

Referrer or Referring Resource: This contains the URL of the document issuing the link to the current request page.
The below figure shows the example of Web Server Log Record Format.

Example:

# IP Address Userid Time Method/ URL/ Protocol Status Size Referrer Agent

1 <123.456.78.9 - [25/Apr/1998:03:04:41 -0500] "GET A.html HTTP/1.0" 200 3290 - Mozilla/3.04 (Win95, I)>

2< 123.456.78.9 - [25/Apr/1998:03:05:34 -0500] "GET B.html HTTP/1.0" 200 2050 A.html Mozilla/3.04 (Win95, I)>
3 <123.456.78.9 - [25/Apr/1998:03:05:39 -0500] "GET L.html HTTP/1.0" 200 4130 - Mozilla/3.04 (Win95, I)>

Figure.3 Example of Web Server Log Record Format

This information can be used to reconstruct the user Identification and user sessions within the site from which the log data originates. In an ideal scenario each user is allocated a unique IP address whenever he accesses a given web site. Moreover, it is expected that a user visits the site more than once, each time possibly with a different goal in mind. Therefore, a user session is usually defined as a sequence of requests from the same IP address such that no two consecutive requests are separated by more than minutes, where is a given parameter. According to the W3C [11] the session or "visit" the group of activates performed by the user from the moment she enter the site and she left the site. Since a user may visit a site more than once, the Web server log records multiple sessions for each user. We use the name \user activity log" for the sequence of logged activities belonging to the same user. Thus, "sessionizing" is the process of segmenting the user activity log of each user into sessions. A tool that implements this process is a "sessionizing heuristic": it reconstructs a session on the basis of assumptions about user behavior. The contents of a (re)constructed session depend on the requirements of the mining application. In many applications, including market basket analysis and establishment of usage problems, the expert is interested in the pages being accessed during a session but not in the order of access, nor on revisitations. Hence, a session is a set of activities. If the navigational behavior of users is studied, the order of access is of interest. Then, a session must be modeled as a sequence of activities. If the effect of revisitations is of interest, e.g. to investigate the causes of disorientation, then it is necessary to expand each session with the pages revisited but not recorded, due to caching. A "real session" contains the activities that the user performed together according to a reference model, which in our experiments is provided by the Web server of the test site.

The below figure shows the preprocessing tasks of Web Usage Mining. The inputs to the preprocessing phase are the server logs, site files. The outputs are the user session file, transaction file

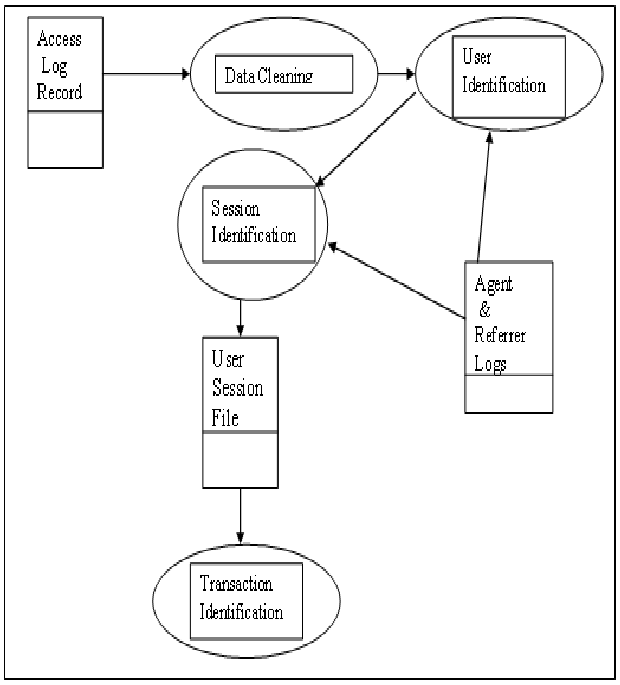

Figure 1.4. The steps in the Preprocessing phase

Access Paths and Patterns

The output of the preprocessing step is user access paths. The below example shows the difference between user access paths and user access patterns [13].

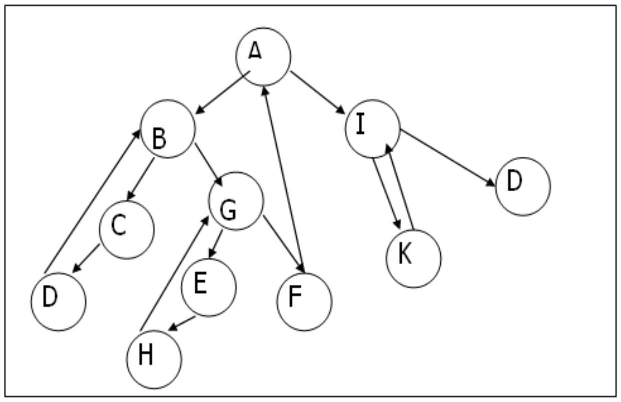

Figure 1.5. Graph Representation of User Access Paths

The figure shows the one user browsing web pages for single session. In this figure the nodes represent the web pages and links represents the navigation of web pages.
The user access patterns follow only forward references.
The user access paths follow both forward and backward references of web pages.

The user path through which Uid accesses certain

website: A-B-C-D-B-G-E-H-G-C-A-I-K-I-D.

The user access patterns are: A-B-C-D, A-B-G-E-H, A-I-K-I-D.

The frequent access patterns have to satisfy the condition i.e. the number of occurrences of access patterns is more than equal to the minimum support. The minimum support is number of occurrences per total transactions. The minimum support also given as input for generating user access patterns.

II EXISTING ALGORITHMS

A. Apriori Algorithm

This algorithm has two important and common steps: (i) Candidate generation, and (ii) I/O operation. The initial candidate set generation, especially for the frequent 2-patterns, is the key issue to improve the performance of frequent pattern discovery algorithms. Another performance measure is the amount of data that has to be scanned during the discovery of the Frequent Patterns [1, 2].

The Apriori algorithm requires one pass over the database of all transactions for each iteration [14].

```
Input: An Access paths database S, andminimumsupport threshold min_sup
Output: The complete set of Frequent Access Patterns.

Algorithm:
Procedure AprioriAlg()
begin
L1:= {frequent 1-patterns}; // find frequent 1-patterns by scanning total database
for ($k$: = 2; $L_{k-1}$ >0; $k$++) do
 begin
  for each pattern l1 belongs to Lk-1
   for each pattern l2 belongs to Lk-2
     $C_k$ = apriori-gen ($L_{k-1}$); // new candidates
  for all user access paths t in the dataset do
  begin
  for all candidates Ck contained in t do
       c:count++
  end of for loop
$L_k$= {c $C_k$ | c:count >= min-support}
end of the for loop
Answer: = k $L_k$
end of the function
```

Figure 2.1. Apriori Algorithm

B. Partition Algorithm

The Partition algorithm attempts to reduce the I/O operations by considering smaller segments of the database [14]. The partition algorithm is based on the observation that the frequent patterns are normally very few in number compared to the set of all patterns. As a result, if we partition the set of transactions to smaller segments such that each segment can accommodated in the main memory, then we can compute the set of frequent patterns of each of these partitions. It is assumed that these patterns (set of local frequent patterns) contain a reasonably small number of patterns. Hence, we can read the whole database (the unsegmented one) once, to count the support of the set of all local frequent patterns

```
Input: A Access Paths database S, and minimum support threshold min_sup
Output: The complete set of Frequent Access Patterns

Algorithm:
Phase I:
 Partition()
begin
  m= find the total memory size;
 M= find the total database size
 N= M /m; // number of overlapping partitions

For I = 0 to N do // for all partitions
begin
 Lk = find frequent patterns of partition I;
 L= L U Lk
end.
end of function partition.

Phase II :
Partition1 ()
begin
 G = global minimum support
  R= {}// represents the global frequent patterns
 //Scan the database for generating global frequent patterns
for I = 0 to N do
begin
 if Lk > G
 R = R U Ri
End for loop
End of the function partition1
```

Figure 2.2. Partition Algorithm for generating Frequent Access Patterns

C. FP-Tree Algorithm

The recent development of the FP-tree algorithm avoids the candidate generation steps [15]. The main idea of the algorithm is to maintain a frequent pattern tree (FP-Tree) of the database. It is an extended prefix-tree structure, storing crucial quantitative information about frequent sets. The tree nodes are frequent items and are arranged in such a way that more frequently occurring nodes will have a better chances of sharing nodes than the less frequently occurring ones. The method starts from frequent 1-itemsets as an initial suffix pattern and examines only its conditional pattern base (a subset of the database), which consists of the set of frequent items co-occuring with the suffix pattern. The algorithm constructs the conditional FP-tree and performs mining on this tree

```
Input: A Access Paths database S, and minimum support threshold min_sup
Output: The complete set of Frequent Access Patterns.

Algorithm:
Phase 1:
Procedure FPTree ()
begin
  Create_tree (T); //construct the root of FAP-Tree signed with "null"
  While (P<>null)
begin
    If (p.name is the same as the name of T's ancestor (n))
     begin
       Increment c.count;
       T=n;
     end of if
    else
      If (p.name is the same as the name of T's child (e))
        begin
          Increment c.count;
          T=c;
        end of if statement
      else
        begin
          Insert_tree (T, p);
           //insert the new node of P into T, as a child of the current node
          P=p.next;
        end of Else statement
  end of Else statement
```

Figure 2.3. FP- Tree Algorithm

III IMPROVED ALGORITHM

There are many existing algorithms for generating frequent access patterns from the access paths. But they have less efficient in terms of execution time and memory requirement. This proposed algorithm is modification of FP-tree Algorithm, but this algorithm will not use recursion for generating Frequent Patterns. So this Algorithm will take less execution time for access paths which are not having uncommon items. This is explained in the below example.
The main idea of the algorithm is to maintain a frequent pattern tree of the database. It is an extended prefix-tree structure, storing crucial quantitative information about frequent patterns. This algorithm is not using recursion unlike FP-tree Algorithm. This algorithm scans the data base once for generating page table. This table stores the information about web pages, the number of times the user accessed that web page and the pointer field that stores the reference of that webpage in the pattern base tree. The page table nodes are sorted according to the page count. The tree nodes are frequent items and are arranged in such a way that more frequently occurring nodes will have a better chances of sharing nodes than the less frequently occurring ones. The method starts from frequent 1-itemsets as an initial suffix pattern and examines only its conditional pattern base (a subset of the database), which consists of the set of frequent items co-occurring with the suffix pattern. The page table nodes are used for generating frequent access patterns. Start from the page table seqptr, which stores the reference of the tree node then traverse the tree from bottom to the root node. Add the entire nodes which are in the traversal with the condition *pagecount* > *min_sup*. If this condition is not satisfied then move to the next path in the tree. Generate all the frequent patterns of the users by using backward traversals of the tree.

Example

Assume A,BC,D,E,F,G,H,I,K are the web pages in a particular web site. U1 is the userID and S1, S2, S3, S4 are different sessions of that particular user. This is shown in the below table.

Table 3.1 Access paths of single user in different sessions

User Name	Session Name	Access Path
U1	S1	A-B-C-D-B-E
U1	S2	A-B-C-D-C-B-E-G-E-C-H
U1	S3	A-B-A-C-I
U1	S4	C-I-G-I-K-I-D

The below figure shows Page table count and Tree generation for all user access paths.

Table 3.2 Access paths of single user

PAGE NAME	COUNT	NEXT
H	1	
K	1	
G	2	
D	3	
E	3	
A	4	
I	4	
B	5	
C	6	

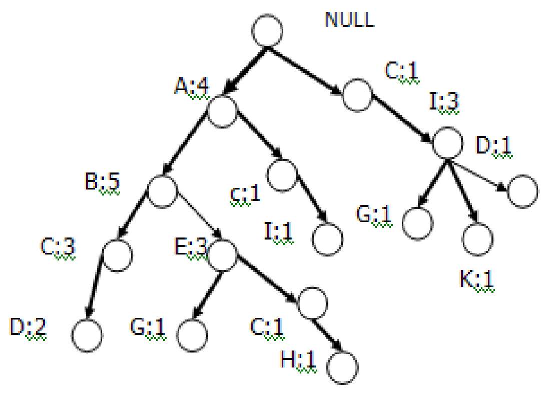

Figure3.3. Tree generation from access paths

In the above page table gives information about web pages and no of occurrences of all the access paths. The page table contents are stored in ascending order according to the user access page count. For generating Access patterns it will take one more database scan. The tree stores all access paths in compressed format. Each node in the tree represents the page name and page count. Each node has two pointers to parent node and to the child nodes. The root node is null node; the child nodes of this root node will give all access paths. If the user follow the same path in different session, this tree simply increment page count instead of creating new nodes.

IV.IMPLEMENTATION DETAILS

Access log record: Access log record is in the web server. This data is stored in text format and it is in unstructured format. First load the data into Data base from the text format.

Data Cleaning: Elimination of the items deemed irrelevant can be reasonably accomplished by checking the Suffix of the URL name. For instance, all log entries with filename suffixes such as, gif, jpeg, GIF, JPEG, jpg, JPG, and map can be removed. Scripts such as "count.cgi" can also be removed.

Step1:- write query for eliminate gif, jpeg, GIF, JPEG, jpg, JPG from the database.

Remove all the rows which are having the above suffix in the Request field in the database.

Step2: By using StringTokanizer class in java divide the agent field into two fields i.e Browser, its version and Operating System.

Step3: Remove irrelevant information in the data field i.e. in the request field. We want user requested page only, but the request is in the below form.

GET/biblio/riviste/img/r-t/rew4.html.

From this we have to remove prefix of rew4.html and again store this rew4.html in Request field of database. Now the data base is useful for the Preprocessing.

User Identification

Step1: First create the website topology**.**

Take each page as one node and link to the previous page and next page. The below java class represent the Webpage.

```
 Public class Webpage   {
    Protected String pagename;
   Protected TreeNode parent; //parent node
   Protected TreeNode fchild; //first child
   Protected TreeNode subling; //gives the subling node }
```

V. RESULTS

For validation of the algorithm data used from the web site *www.musicmachines.com*. The log records are available from September 1998 to December 1998. Simulations were performed using an AMD Athlon processor, with 256 MB of main memory, 756 MB of virtual memory, 40 GB of local disk space and on Microsoft Windows XP Operating System. These results checked for constant size data base (i.e50MB, 150MB). The two algorithms i.e. Apriori Algorithm and Proposed Algorithm are implemented by using Java.

The user screens are shown in the Appendix A for both the algorithms.

The following figure shows the comparison result.

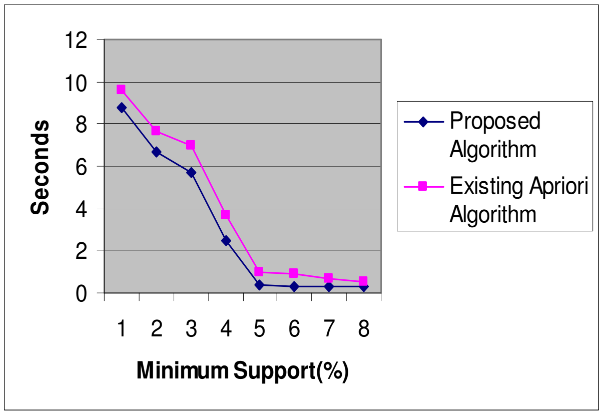

Figure 3.4. Apriori and Proposed Algorithm results Comparison

Analysis Of Results

The above figure shows the comparisons of both the algorithms in term of execution time. An evident from the figure, the minimum support is less then the execution time is more; because more number of candidate sets will be generated. Hence, Apriori Algorithm execution time is more than the proposed Algorithm. The proposed Algorithm is not generating any candidate sets, but more number of patterns will be generated, due to this the number of tree traversals will be more. From the above Figure 3.4 the proposed algorithm is taking less time compare to the Apriori algorithm in all instances. If the minimum threshold is more the both algorithm execution time is less.

VI.CONCLUSION AND SCOPE FOR WORK

Information content on the WWW is increasing at an exponential rate and it is not surprising to find users having difficulty in navigation and finding relevant information. Hence, the e-commerce site developers find it difficult to observe potential customers or web site structure. This paper used a Web Access log file of a Web site to apply data mining techniques for finding frequent access patterns of the users.

VII.SCOPE FOR WORK

However the work may be extended to analyze the Data Preprocessing phase in detail. One work has been carried out mostly for the frequent pattern analysis. The work can be extended to analyzed and suggest modifications for the Data Preprocessing phase. It can also simulated using variable memory sizes, instead of the constant memory sizes, instead of the constant memory size adapted for the study. Graph theory and Statistical analysis etc. can also be done for Web Usage Mining. By using efficient algorithm we can reduce the runtime and memory requirement

VIII.REFERENCES

[1] Wang Jicheng, Huang Yuan, Wu Gangshan and Zhang Fuyan. Web Mining: Knowledge Discovery on the Web. IEEE Intl Confirence, 1999, p 137-141.

[2] Yan Wang. Web Mining and Knowledge Discovery of Usage Patterns, Project (part1), CS748T, February, 2000, Worcester polytechnic Institute.

[3] Jiawei Han and Jai pei. Mining Frequent Patterns by pattern Growth: Methodology and Implications .SIGKDD, December2000, ACM.

[4] Lizhen, Junjie Chen and Hantao Song. The Research of Web Mining, the 4th World Congress on Intelligent Control and Automation. Pp 2333 – 2337

June 10 -14, 2002.

[5] R. Cooly, B. Mobasher and J. Srivastava. Web Mining: Information and Pattern Discovery on the World Wide Web. Pp.558 – 566. August 1997 IEEE.

[6] Jaideep Srivasthava. Web Mining accomplishments and Future Work. http:// www.cs.umn.edu/faculty/srivasta.html visited on november 2004.

[7] Yan Wang. Web Mining and Knowledge Discovery of Usage Patterns, Project (part1), CS748T, February, 2000.

[8] R. Cooley, B. Mobasher and J. Srivastava. Data Preparation for Mining World wide Web Browsing Patterns. KISS, October 1998.

[9] http://www.webtrends.com/WUM.html

[10] R. Cooley, B.Mobasher and J. Srivastava. Grouping Web Page References into Transactions for Mining World Wide Web Browsing Patterns. Dec2000, IEEE.

[11] www.w3c.org/clf.html

[12] Gabriele Bartolini. Web Usage Mining and Discovery of association rules from HTTP server logs. October 2001,IEEE.

[13] Jain Pei, Jiawei Han, BehZad Motazavi-asl and Hua Zhu. Mining Access Patterns Efficiently from Web Logs. Technical ReportCS2000, CS, imon Fraser University.

[14] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In Proc. of Intl. Conf. on Very Large Databases (VLDB), September 1994.

[15] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In, pp-365-378, September 1998, IEEE.

[16] J. Han and M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, San Francisco, CA, 2001.

[17] Fang Zhang and HuI-You Chand Research and Development in Web Usage Mining Systems—Key Issues and Proposed Solutions. A Survey. IEEE International Confirence, pp.986, April 2002.

[18] Lizhen Research, Chen and Hartao Song. The Research of Web Mining, The 4 th world Congress on Intellegent Control. Pp 238 -242, January 1998, IEEE.

[19] IBM, http://www.software.ibm.com/data/iminer

[20] Jaideep Srivasthava. Web Mining accomplishments and Future Work. srivsta@cs.umn.edu http:// www.cs.umn.edu/faculty/srivasta.html

[21] WangBin and LiuZhijing. Web Mining Research, Fifith International Conference on Computational Intelligence and Multimedia Applications, Jan 2003.

[22] www.mysql.org/tutorial

[23] Park Jong Soo, Chen Ming-Syan, and Yu Philip S. Using a hash-based method with transaction trimming for mining association rules. IEEE transactions on knowledge and data Engineering, 9 ,no. 5,Sept/oct 1997.

[24] www.musicmachines.com/downloads/sampledata

[25] F. Masseglia, P. Poncelet and M. Teisseire. Using Data Mining Techniques on Web Access Logs to Dynamically Improve Hypertext Structure. In *ACM SigWeb Letters*, 8(3):13-19, October 1999.

[26] T. W. Yan, M. Jacobsen, H. G. Molina, and U. Dayal. From User Access Patterns to Dynamic Hypertext Linking. In *Proceedings of the 5th International Wrold-Wide Web Conference*, pages 7-11, Paris, France, May 1996.