ISSN No. 0976-5697

Volume 2, No. 1, Jan-Feb 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Web Usage Mining using Improved FP Tree Algorithm with Customized Web Log Preprocessing

Prateek Gupta*
Comp. Sc. & Engg.,
RGPV , SSSIST
Sehore, Madhya Pradesh , India
prateek_guptace@hotmail.com

Surendra Mishra
Comp. Sc. & Engg.,
RGPV , SSSIST
Sehore, Madhya Pradesh, India
smishra1lnct@gmail.com

Abstract: Web Usage Mining mines Web access logs for interesting patterns in WWW traffic. Web Usage Mining discovers interesting patterns in accesses to various Web pages within the Web space associated with a particular server. The Web Usage Mining architecture divides the process into two main parts- the first part includes preprocessing, transaction identification, and data integration components. The second part includes the largely domain independent application of generic data mining and pattern matching. Input to the Web usage Mining process is Web Server Logs, Error logs, User Cookies and Client Cache Records. This paper contains an efficient improved iterative FP Tree algorithm for generating frequent access patterns from the access paths of the users. The frequent access patterns are generated by backward tree traversals. This operation will take less time compare to the existing algorithms. Paper also composed of customized web log preprocessing for mined in different applications.

Keywords: Web Usage Mining, FP Tree, Web Logs, Web Log Preprocessing, Customized Web Log Preprocessing

I. INTRODUCTION

The Web Mining [1] is the application of Data Mining techniques to automatically discover and extract information from the web. Web usage mining is the task of discovering the activities of the users while they are browsing and navigating through the Web. The aim of understanding the navigation preferences of the visitors is to enhance the quality of electronic commerce services (e-commerce), to personalize the Web portals [2] or to improve the Web structure and Web server performance Web usage mining [3], from the data mining aspect, is applying data mining techniques to discover usage patterns from Web data. Examples of applications of such knowledge include improving designs of web sites, analyzing system performance as well as network communications, understanding user reaction and motivation, and building adaptive Web sites.

The process of Web usage mining also consists of three main steps: (i) preprocessing, (ii) pattern discovery and (iii) pattern analysis.

In this work pattern discovery means applying the introduced frequent pattern discovery methods to the log data. For this reason the data have to be converted in the preprocessing phase such that the output of the conversion can be used as the input of the algorithms. Log files are stored on the server side, on the client side and on the proxy servers. Logs are processed in Common Log Format. Pattern analysis means understanding the results obtained by the algorithms and drawing conclusions. In pattern discovery phase methods and algorithms used have been developed from several fields such as statistics, machine learning, and databases. This phase of Web usage mining has three main operations of interest: association (i.e. which pages tend to be accessed together), clustering (i.e. finding groups of users, transactions, pages, etc.), and sequential analysis (the order in which web pages tend to be accessed). Pattern analysis is the last phase in the overall process of Web usage mining.

In this phase the motivation is to filter out uninteresting rules or patterns found in the previous phase.

If all the transactions are different i.e. no common access paths then Apriori algorithm is good otherwise FP-tree Algorithm is good [4]. Apriori Algorithm will take more time and more memory compare to the FP-tree algorithm. The partition approach is good if data base size is very large; if database size is less then it will work as Apriori algorithm [6]. In this paper we have used two main tasks-Customized web log preprocessing and Improved FP Tree Algorithm

II. EXISTING WORK

A. Web Log Preprocessing

The inputs to the preprocessing phase are the log and site files. The outputs are the user session file, transaction file Web servers register a Web log entry for every single access they get, in which important pieces of information about accessing are recorded, including the URL requested, the IP address from which the request originated, and a timestamp. A log file can be located in three different places-Web Servers, Web proxy Servers, and Client browsers [8]. The most popular log file format is the Common Log Format (CLF)-

<ip><base url><date><method><file><protocol><code> <bytes> <referrer><user agent>

Preprocessing [9] contains four sub steps: Data Cleaning, User Identification, Session Identification and Formatting.

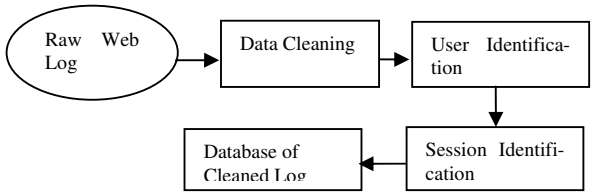

Figure-1 Preprocessing Steps

B. FP Tree Algorithm

The FP-tree algorithm avoids candidate generation steps [7]. The main idea of the algorithm is to maintain a frequent pattern tree (FP-Tree) of the database [5]. 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-occurring with the suffix pattern. The algorithm constructs the conditional FP-tree and performs mining on this tree. A hash-based technique is used to reduce the size of the candidate *k*- patterns. Another variation is to reduce the number of transactions to be scanned at higher values of *k*. Since a transaction that does not contain any frequent *k*-pattern cannot contain any frequent (*k*+1) - pattern, these types of transactions can be marked during the *K*th scanning and are not considered in the subsequent scanning.

III. ANALYSIS OF EXISTING WORK

The existing algorithms are Apriori Algorithm, Partition Based Approach and FP-Tree algorithm, etc. All existing Algorithms have their own advantages and drawbacks. If all the transactions are different then Apriori algorithm is good otherwise FP-tree Algorithm is good. If the user access paths are common then FP-tree algorithm is very efficient. It stores all data in the compressed format. So memory requirement for this algorithm is less compare to the other algorithms. If all the transactions are different the tree will become complex for finding frequent patterns. In this case Apriori Algorithm is more efficient than the FP- tree algorithm. The partition algorithm is efficient for large size data bases. This algorithm uses different algorithms for generating local patterns.

In FP-tree Algorithm, the tree will generate one branch for each transaction. The tree size will become complex for storing in the memory. The FP-tree Algorithm will take more time for recursive calls in the algorithm. For generating Frequent Patterns the pointer have to traverse all the nodes. So it will take more time.

Existing preprocessing method gives output for some particular application, but different application requires different type of preprocessing to get desired output.

IV. PROPOSED WORK

A. Customized Web Log Preprocessing

Different web application requires different preprocessing of logs. Multimedia application requires log of multimedia link request like log having jpg, mpg, and gif etc. resource. All application removes error log. E-commerce application requires different user requests.

We introduced one more step in traditional preprocessing steps, before data cleaning, Customization. In this step we clean log on the basis of user requirement for application.

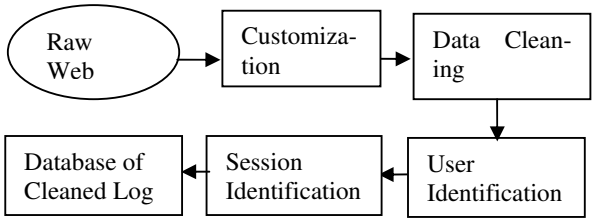

Figure-2 Customized Preprocessing Steps

B. Improved FP Tree 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. The 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.

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 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. This algorithm is divided into two steps:

Step 1: Construct frequent access pattern tree according to access paths derived from user session files, and records the access counts of each page.

Input- an Access Paths database *S*, and minimum support threshold *min_sup*

Output- The Page Header Table, FP- Tree.

Procedure FAP_Tree (*T*, *p*)

begin

Create_tree (*T*);

//construct the root of FAP-Tree signed with "*null*"

While (*P*<>*null*) do

begin

```
    If (p.name is the same as the name of T's ancestor (n))
     begin
          Increment n.count value
          T=n;
          end of if statement
    else
     begin
      If (p.name is the same as the name of T's child (e) )
          begin
              Increment c.count value
               T=c;
           end
        else
            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 the else statement
end
```

Step 2: The function of FAP_growth is used to mine both long and short access patterns on the FAP tree, which is created in Step1.

```
Input: FAP_tree, min_sup = t
Output:  the set of all Frequent Access Patterns: k
Procedure FAP-growth (tree, t);
 //tree is generated tree in step1
begin
    For each Ki.count>=min_sup
          //Ki is a member of the page header table
    begin
         Generate access pattern B=Ki
         K = K U B;
         P= ki.next;
        //p points to the first location of Ki in the FAP-tree
       While (p! =null) and (p.count>=min_sup) do
        begin
        // Look for each Ki's prefix access pattern base,
        //then construct access pattern Bi by Ki prefix
        //access pattern base connecting with itself;
       If (Bi>= min_sup)
          Ki=ki U Bi;
        //adding newly generated patterns in to pattern base
          p=p.next;
        //p points to the next location of Ki in the FAP tree
        end of the while loop
 end of the for loop
end of the function
```

V. 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 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 table shows Page table for all user access paths.

Table 2 : Access paths of single user

Page Name	Count
H	1
K	1
G	2
D	3
E	3
A	4
I	4
B	5
C	6

Following figure represent tree generated from access Path

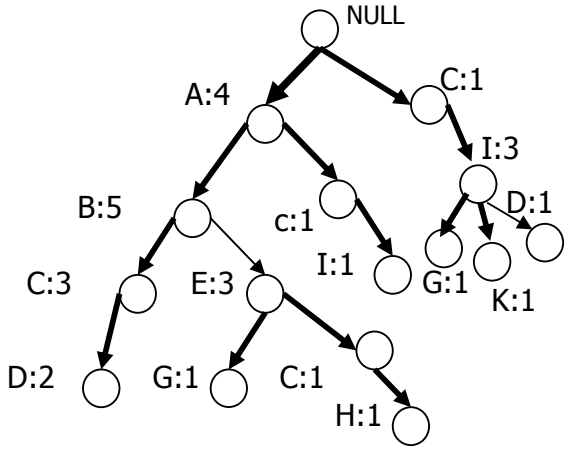

Figure-3 Tree generation from access paths

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.

For generating Frequent access patterns, start from the page table first page traverse to the tree node if it follows the below condition:

```
 If page count>min_support:
begin
 Move header node seqptr to the treenode,
Travers the treenode to the rootnode
Move present node to next seqptr and again follow the above two steps
End
Else
Go to next page.
```

VI. RESULT

The following figure shows the comparison result.

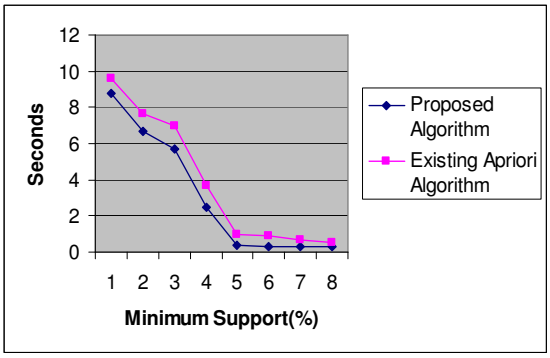

Figure-4 Existing and Proposed Algorithm results comparison

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. 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.

VII. CONCLUSION

Proposed algorithm will take at most two data base scans for generating the frequent access patterns. The maximum tree size required is at most the number of web pages in the website. An even faster response can be obtained if the depth of the tree is increases, i.e. user access paths are increased in the transactions. Further Addition of new paths does not create any update problems, which are very common in existing algorithms. The system is also scaleable using multiple disk storage. This scaling routine takes constant time overhead for memory partition and swapping operations.

ACKNOWLEDGMENTS

"Web Usage Mining using improved FP Tree algorithm, with customized web log preprocessing" research is supported by the management of **SSSIST, Sehore, Madhya Pradesh, India**, We are pleased to acknowledge.

VIII. REFERENCES

[1] R. Cooly, B. Mobasher and J. Srivastava, "Web Mining : In- formation and Pattern Discovery on the World Wide Web" , IEEE, August 1997. Pp.558 – 566.

[2] Charu C. Aggarwal and Philip S. Yu, "An Automated Sys - tem for Web Portal Personalization", Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002

[3] Renáta Iváncsy, István Vajk, "Frequent Pattern Mining in Web Log Data", Acta Polytechnica Hungarica, Vol. 3, No. 1, 2006

[4] B.Santhosh Kumar, K.V.Rukmani, "Implementation of Web Usage Mining Using Apriori and FP Growth Algorithm", Int. J. of Advanced Networking and Applications, Volume:01, Issue:06, (2010), Pages: 400-404

[5] Shui Wang, Le Wang, "An implementation of FP growth algorithm based on high level data structure of Weka-JUNG framework", Journal of Convergence Information Technology, Volume 5, Number 9, 2010

[6] Kotsiantis S, Kanellopoulos D., "Association Rules Mining: A Recent Overview", GESTS International Transactions on Computer Science and Engineering, Vol.32 (1), 2006, pp.71- 82

[7] J. Han, J. Pei, and Y. Yin. "Mining frequent patterns with – out candidate generation". IEEE, Sept.1998 pp-365-378.

[8] K. R. Suneetha and Dr. R. Krishnamoorthi, "Identifying User Behavior by Analyzing Web Server Access Log File",IJCSNS International Journal of Computer Science and Network Se-curity, VOL.9 No.4, April 2009, pp. 327-332.

[9] P. Brusilovsky, A. Kobsa, and W. Nejdl (Eds.): The Adap - tive Web, LNCS 4321, 2007, pp. 90–135,.

AUTHORS

Prateek Gupta received bachelor degree in Computer Science & Engineering from Rajiv Gandhi Proudyogiki Vishwavidyalaya, State Technological University of Madhya Pradesh and pursuing Master of Technology in Computer Science & Engineering from SSSIST, Sehore under Rajiv Gandhi Proudyogiki Vishwavidyalaya. He has publish many papers on data mining and participated in many workshops.

Surendra Mishra Master of Technology in Computer Science & Engineering from SSSIST, Sehore under Rajiv Gandhi Proudyogiki Vishwavidyalaya, State Technological University of Madhya Pradesh. He is working as an Asst. Professor and HoD Comp. Sc. & Engg. In SSSIST, Sehore, Madhya Pradesh. He has published many papers on data mining.