An Approach for Frequent Access Pattern Identification in Web Usage Mining

Murli Manohar Sharma
Computer Science and Engineering Department
Thapar University
Punjab, India
sharmamurli004@gmail.com

Anju Bala
Computer Science and Engineering Department
Thapar University
Punjab, India
anjubala@thapar.edu

***Abstract*—- In the consent of this internet world, nobody is untouched with internet for their usage. For such kind of scenario, data mining becomes an essential part of computer science. Data mining is a sub-field, which computationally processes the data collected data and is able to help the analyst for proposing the ideas for some betterment of the company. The user access is recorded in log files. The web server logs provide important information. In the field of web mining the analysis of the web logs is done to identify the user search patterns. In the usual approaches of finding the patterns, pattern tree is created and then the analysis is done, but in this proposed algorithm there is no need of tree creation and the analysis is done based on the website architecture, which will increase the efficiency of the other pattern matching algorithms and needs only one database scan.**

Keywords—* *Web usage mining, Maximal forward references, large references, Algorithm

I. Introduction

In the current scenario of the world wide web, the popularity is increasing day by day, so as the web mining. With the increasing number of websites and web users, the data of the web usage is stored on the web servers. By the analysis of this web server data, we can have various information like user surfing behavior which is the most important aspect of business marketing and can help us user profiling, web site designs meliorate and making better business and marketing decision making our website more popular and user friendly. But by looking at a small set of data we cannot identify the patterns [1], for this purpose the analyst need a significantly large amount of data. All the data collected by the service providers is stored in the high capacity servers. As the user becomes high in the numbers this data also grows and the data logs are not easy to maintain. To analyze such kind of large data we need to extract the useful data and this data is then mined to get the patterns of the user behavior. For this purpose an efficient algorithm is needed, which can do the purpose and help extracting the information. There are so many algorithms which are liable to resolve the purpose but they all take the time in scanning and pattern matching. In this paper, an algorithm is designed which employs the website architecture and gives the information about the users' usage behavior. The users access a website by going from one page to another page, by the hyperlinks provided in the web pages. Mining the information identified by the analysis, will not only help making the user interface better but also in various business decision making.

The field of data mining and the user behavior has become a necessity. The user can traverse a web-site in different-different ways. The variations between traversal patterns increase the complexity of obtaining information from the path traversed. There are various algorithms available for citing the user traversal patterns. In this paper a new approach has been proposed for mining the large reference patterns. The traversal patterns are achieved first by mining the maximal forward references from the web server log and after this the maximal forward reference are obtained and the large references can be calculated, which are the most frequently used paths followed by the user for a particular website.

This paper basically contains a proposal of a new algorithm for getting the reference counts by first obtaining the maximal forward reference using the algorithm MF for obtaining traversal subsequences from the log file data and later on apply a new algorithm to obtain the frequently occurring traversal patterns.

II. Literature Survey

Oren Etzioni [2], first proposed the term of web mining in 1996 in his paper. In this paper he claimed that web mining is the use of data mining techniques to automatically discover and extract the information from World Wide Web documents and services. In the same paper, Etzioni came up with the question: Whether effective web mining is feasible in practice? Today, with the tremendous growth of the data sources available on the web and the increased popularity of e-commerce in the corporate world, web mining has become the main ingredient of quite a few research projects and papers.

Basically, data mining is a collection of technique of efficient automated discovery of patterns in large databases. These patterns are actionable so that they may be used in an enterprise's decision making processes. Data mining and the user behavior recognition is used by the business intelligence organizations, financial analysts and the research and development department, but it is being used in the sciences to extract information from the large data sets generated by modern experimental and observational methods also[3]. Now moving on to the problem statement and this paper focuses on a new approach for identifying the user traversal patterns, along with a short overview of the existing algorithms.

A. PROBLEM DESCRIPTION

The World Wide Web is made up of a large amount of information which is in a web structure where the web-pages are considered as nodes and they are linked with each other via hyperlinks represented by the arrows. The users can move forward and backward with the help of these hyperlinks. Some pages may be visited again by the user because of the placement of the web pages and its contents, say for example to visit a sibling pages the user usually uses the backward icon and then forward icon instead of going straight to the URL.

So finally to get useful traversal patterns of the user traversals from the server data the need is to avoid these backward references and extract the useful patterns. The backward traversals are just for for the ease of moving to the previous page and the main concentration is on the discovery of meaningful forward user access patterns. The path the user was moving on terminates, when a previous access is revisited by the user. This gives a forward reference path which is termed as maximal forward reference (MFR)[4]. After obtaining the maximal forward reference it starts again from the starting point from where the forward reference path was being started to be obtained and then a new user traversed path is obtained. Also if a null source node appears termination should be made and a new path should be found again.

As it is known that the web pages are represented as a nodes and the hyperlinks are the links between these nodes, the metadata of the user traversal is stored in the log files. Suppose the log contains various user traversed path say : {A,B,C,D,E,F,G,H,I,J} as in Figure 1.

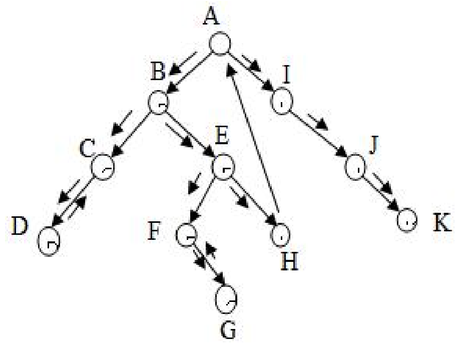

Fig. 1 Path Traversed by the User

Now, by using the maximal forward reference algorithm, the forward references obtained are: {ABCD, ABEFG, ABEH, AIJK}. After obtaining the maximum traversed patterns the frequently occurring substrings from the maximal forward references are identified. The frequently occurring substring is large reference. A large reference sequence is a traversal pattern which appears more number of times.

The procedure can be summarized as follows

Step 1: Maximal Forward Reference is identified by MF algorithm.

Step 2: Extract the large reference sequence by using the proposed algorithm.

III. ALGORITHMS FOR USER ACCESS PATTERNS

Maximal forward references[5] are those strings which tell the maximum length till which the user has accessed a particular website before it stepped back to a previously accessed node (webpage). A server log consist of various fields as hostid, username, date and time, request query, current status of the transaction, bytes transferred, user-agent. Now, a pair of source and destination is captured from the server log file. This pair can help us getting the maximal forward references. The server log database file is sorted based on the user id's where the (source, destination) pair is ordered by time. The output obtained from MF is stored in a database DF as this output will act as input to the next algorithm for reference scan, which gives us the large reference sequence.

ALGORITHMS FOR FINDING MAXIMAL FORWARD REFERENCES-

Algorithm MF [6]

```
Step 1- Set $a$ = 1 and string $S$ = null at the time of initialization,
        where string $S$ holds the current forward reference
        path.
        Set a flag $F$ to 1 to indicate a forward traversal.
Step 2- Let $A = si$ and $B = di$.
            If $A$ is equal to null the
            /* new traversal starts*/
          Begin
            Write the ongoing string $Y$ (if not null) to the
            database $DF$;
            Set string $S = B$;
            Go to Step 5.
          end
Step 3- If $B$ is equal to some reference (say the j$^{th}$ reference) in
        string $S$ then
/* going back to the previously referenced page */
          begin
            If $F$ is equal to 1 then write out string $S$ to
            database $D_F$;
                Discard all the references after the j$^{th}$ one in
            string S
            F =0;
          Go to Step 5.
        end
Step 4- Otherwise, append B to the end of string S.
        If F is equal to 0 then set F = 1.
Step 5- Set a= a+l. If the sequence is not completed scanned
        then go to Step 2.
```

A. Existing Algorithms

Various algorithms have been proposed to obtain the large references, which work on the technique of hashing and cropping. The algorithms proposed were 1)Full Scan, 2)Selective Scan, 3)Reference Scan, 4)FP-Tree algorithm. These algorithms work on the concept of data hashing and pruning, which involves scanning the database or the log files and this database is cropped in each scan. The brief explanation of these algorithms is given as follows.

1) Algorithm on Full scan (FS)

Let L_k be the set of all large k-references and C_k is a candidate-set with k-references. C_k is usually a superset of L_k**.** By scanning the whole database D_F**,** FS gets L_1 and makes a hash table (i.e., H_1*)* to count the number of occurrences of each 2-references.

In FS, starting with k = 2, FS generates C_k**,** based on the hash table count obtained in the previous pass, determines the set of large k-references, reduces the size of database for the next pass, and makes a hash table to determine the candidate *(k* + 1)-references [7].

From the set of maximal forward references, among k-references in Ck, large k-references are obtained. After the scan of the entire database, those k-references in ck with count exceeding the threshold become L_k**.** If L_k is non-empty, the iteration continues for the next pass, i.e., pass k + 1. Every time when the database is scanned, the database is trimmed by FS to improve the efficiency of future scans.

2) Algorithm on selective scan (SS)

Algorithm SS works the way FS does, it also employs hashing and cropping techniques. These techniques are used to reduce both CPU and 1/0 costs, but is different from the latter in that algorithm SS, by properly utilizing the information in candidate references in prior passes, is able to avoid database scans in some passes, thus further reducing the disk 1/0 cost [8, 9].

These algorithms scan the databases to create a candidate table (C_1,C_2,C_3…C_n) and after that by trimming the value with minimum support it obtains the large reference table (L_1,L_2,L_3…L_k). Now the second candidate table is obtained by making the combinations of that table items of large reference table obtained in step 1 and the occurrence of the two item set in main table is considered. After this removal of the items having the least minimum support is done so as to obtain the trimmed database. The further procedure involves generation of next item set and so on. This algorithm uses hash table to store the values. In full scan the database is scanned in each step so as to generate both candidate and large reference table but in full scan the database scans are reduced so as to produce the large references in batches.

3) Algorithm for reference scan [8]

As it is known to us that the website is built in a hierarchical structure starting with the initial webpage and then the rest of the webpage's are connected via the hyperlink. The information mostly lies on the last level of the tree structure with the previous levels containing the folders and the sub folders. Keeping this fact in mind an algorithm has been devised known as reference scan in which the last level is from where we start and move towards the upper levels so as to get our large reference sequences. The algorithm is reducing the database scan and is making our search more efficient and fast. The proposed algorithm is presented below.

```
Algorithm[10]
Input -  Array of structure of TID, seq, min
         support value
Output - Large reference

 1. Find max i.e. maximum length of seq from inputs.
 2. Repeat for flength from max to 1
 3. Find number_of_sequences from flength
 4. if(number_of_sequences<minimumsupport)
 5. continue;
 6. else
 7. Create subsets of all web-pages in forward direction
    of length equal to flength.
 8. Compare all subsets with all input to find occurrence
    of subsets
 9. Get subset with maximum occurrence and
    occurrence>minimum support
10. End if-else
11. End loop
```

4) Fp-growth alforithm

In the above algorithms like Full Scan and Selective Scan (these both are the categories of apriory algorithm), the bottleneck was the candidate set generation. These problems can be overcome by using the FP-growth algorithm [11].

Let I = {i_1,i_2,….. i_m} be a set of items, and a transaction database DB = {b_1, b_2, …. b_n}, where b_i (i is from 1 to n) is a transaction which contains a set of items in I. The support of a pattern A, which is a set of items, is the number of transactions

containing A in DB. A, is a frequent pattern if A's support is no less than a predefined minimum support threshold, i_r.
Given a transaction database DB and a minimum support threshold, i_r, the problem of finding the complete set of frequent patterns is called the frequent pattern mining problem. Now the problem is to find all frequent item sets or frequent patterns in the following database using FP-growth algorithm.

Table 1 Database Input Table

TID	Item
1	E,A,D,B
2	D,A,C,E, B
3	C, A, B, E
4	B, A, D
5	D
6	D, B
7	A, D, E
8	B, C

This is the table which shows the transaction id as TID and the pages that are traversed by the user. After getting this table the priorities are to be assigned according to the reference counts. Once these nodes are arranged in their priorities, the paths are taken into consideration one by one and a tree is formed. Each path starts from a root node say ‘null’. The traversals which doesn’t start with the null node, the null node is pre attached to the path and the counts are stored. Once this is done, the tree is obtained which is shown in Figure 2.

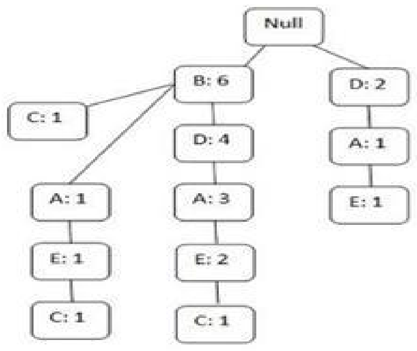

Fig. 2 Final FP-Tree

This tree shows the frequencies of the web pages traversed by the user. The number in the node shows that how frequent the user tried to access.

IV. PROPOSED WORK

In this proposed work, the website structure itself is used during the analysis. The website structure means that the connectivity of the web pages that constitute the web data. In this algorithm, this architecture is considered only, in the form of the tree. After this
This algorithm takes an array named node_path_table as an input. The basic entity must be as follows.

```
struct
{
   Char x; //for transaction id
   Char  y[10];  //for storing the items
}
```

SSPRA (Single Scan Pattern Recognition Algorithm)
Input: Database, Website Structure, node_Path_table, s=a variable, x=tid of the node_path_table

```
1) Foreach transaction in database d.
      a. If Transaction t is not starting from the root
            i. Get the first node of the transaction
               t, in s
           ii. Repeat the node_path_table until
               s=node_path_table[i].x
          iii. Prefix node_path_table[i].y to the
               transaction t.
   //now we have to work on the counts
2) Foreach node in the transaction
      a. Increase the count value of node by 1
3) Get the nodes with the maximum counts and the
   preceding sequence form the node_path_table.
```

Now each node in the tree is equipped with the corresponding count values. Next there is a need to identify the most frequent path. For this purpose that path is chosen which is having the sum of the count values as the maximum. This process will take only one scan.
Because this scans the database only once it may be named as Single Scan Pattern Algorithm. Let us suppose that we have the website structure as given in Table 2. Here the transaction id of the user transaction and the path followed by the user is given.

Table 2 Input Data

Tid	Database
100	ABCD
200	ABEFG
300	ABEH
400	AIJK

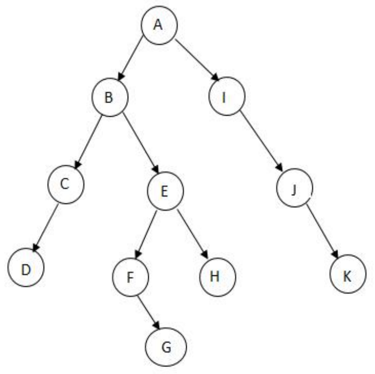

Fig. 3 Website Structure

In Figure 3 the website structure is shown. This shows the way, the web pages are connected to each other. Now by analyzing the transactions we can get the Table 2. This table shows the transaction id and the path that the user followed. In this algorithm, the first transaction is taken and then that path is followed. Each node have 0 as a count. This count value is increased by one, whenever it is traversed in a path.

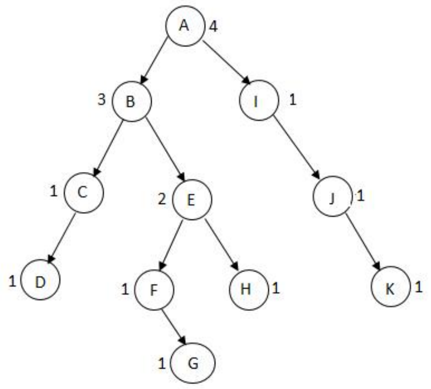

Fig. 4 When the Count is Completed

Figure 4 shows the situation when all the paths are identified and the reference count of each node. Here the numbers shown are the reference counts based on this, it can be identified that user traversed what kind of products and what kind of improvements are needed in the business layouts, or may recommend users for any suitable purchase.

V. Conclusion

As it can be seen that while determining the large references form the reference count table, the apriori algorithms (Full Scan and Selective Scan) scan the reference tables many times, during frequent pattern generation. Full scans and the Selective scan algorithms go through the database tables n times, where n is the number of transactions. The comparison between the existing algorithms and the proposed one is given in Table 3

Table 3 Comparison Table

Property	Full Scan	Selective Scan	FP-Growth	Proposed Algorithm
Database Scans	n	n	2	1
Tree Creation	Yes	Yes	Yes	No
Candidate Generation	Yes	Yes	No	No

In full scan and the selective scan the database scans are n, where n is the number levels of the tree created. While in case of FP-Tree algorithm these are constants. As the no. of scans are reduced, the need for the execution capacity will automatically less. The concerned graph is Figure 5.

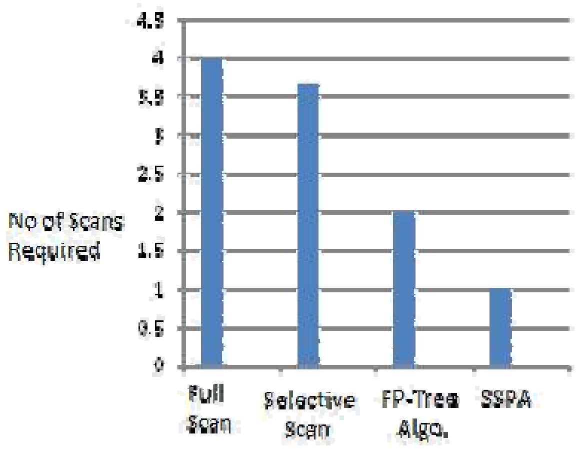

Fig. 5 No. of Scans Needed

Now say no of level in a tree are N and the tree contains M nodes and the no of branches a node may have at a time is X, then we can say that

$$N = \log_x M \quad (1)$$

Where

N: no. of levels in a tree
M: no of nodes in the tree
X: The maximum no. of branches in a tree

This graph tells that in full scan the no of database scans needed are 4. So the no. of nodes in the tree are 81, the levels are 4 and the maximum branches a node can have are 3. In case of selective scan the levels in the tree are 3.67 (about 4), the no. of

nodes are 13 (approx.) and the maximum allowable branches that a node may have are 2. For the FP tree algorithm and the SSPA algorithm the no. of scans are constant i.e. 2 and 1 respectively.
There is no need to think that FP-tree algorithm is far better than any other previous designs, but in this SSPA algorithm no tree is created. Creating an FP tree becomes an overhead sometimes, especially when there is large sized web structure. But in this proposed algorithm only the web structure is used and we can mine the data to find out the behavior of the user and help in business decision making.

Reference

[1] W. Xindong, W. Gong-Qing, Z. Xingquan and D. Wei "Data mining with big data." *Knowledge and Data Engineering, IEEE Transactions,* Vol. 26, Issue No.1, pp no. 97-107, 2014

[2] Etzioni, Oren, "The World-Wide Web: quagmire or gold mine?." *Communications of the ACM,* vol. 39, Issue No. 11, pp no. 65-68, 1996

[3] A. Singh and S. Sharma, “Role of Page ranking algorithm in Searching the Web: A Survey”, *International Journal of Engineering & Technology, Management and Applied Sciences*, Vol. 1, Issue 1, June 2014.

[4] P. Bari and P. M. Chawan. "Web Usage Mining", *Journal of Engineering Computers & Applied Science,* Vol. 2, No. 6, pp. 34-38, 2013

[5] R. Sharma and K. Kaur, "Review of Web Structure Mining Techniques using Clustering and Ranking Algorithms", *International Journal of Research in Computer and Communication, IJRCCT, Vol.* 3, No. 6, pp. 663-668, 2014.

[6] H. Fu Li, S. Lee and M. Kwan, “Online mining (recently) maximal frequent item sets over data streams”, *Proceedings of the 15th International Workshop on Research Issues in Data Engineering,* pp no11-18, 2005

[7] C. Tsai, C. Lai and M. Chiang, "Data mining for internet of things: A survey," *IEEE Communication Surveys & Tutorials*, Vol. 16, No. 1, 2014.

[8] M. S. Chen, J. S. Park and P. S. Yu, “Efiicient Data Mining for Path Traversal Patterns”, *IEEE Transaction on Knowledge and Data Engineering,* Vol. 10, Issue 2, pp no. 209-220, 1988s

[9] F. S. Gharehchopogh and Z. A. Khalifelu, "Analysis and evaluation of unstructured data: text mining versus natural language processing" *Application of Information and Communication Technologies (AICT), 2011 5th International Conference on*. IEEE, pp. 1-4, 2011.

[10] C. Kaur, and R. R. Aggarwal, "Reference Scan Algorithm for Path Traversal Patterns.", *International Journal of Computer Applications,* Vol. 48. Pp no. 20-25, 2012

[11] V. Losarwar, M. Joshi, “Data Preprocessing in Web Usage Mining”, *International Conference on Artificial Intelligence and Embedded Systems (ICAIES'2012)*, pp. 15-16, 2012.