International Journal of Contemporary Research in Computer Science and Technology (IJCRCST) e-ISSN: 2395-5325
Volume 2, Issue 11 (November '2016)

A STUDY ON FOL-MINE – FIRST OCCURRENCE LIST MINE

S.Prabhu,
M.Phil. Scholar,
Department of Computer Science,
Sankara College of commerce and science,
Coimbatore, India.

R.Kalaivani,
Assistant Professor,
Department of Information Technology
Sankara College of commerce and science,
Coimbatore, India.

Abstract: A novel Web Access Pattern Mining Algorithm, FOL-Mine is proposed in this chapter. The FOL-Mine algorithm generates access patterns by suffix building in the projected data base of each frequent event and eliminates the need for construction of pattern tree. A data structure, First Occurrence List (FOL), is introduced for efficient handling of suffix building. Rebuilding of projection databases is completely eliminated in the new method. Experimental analysis of the proposed algorithm reveals significant performance gain over other web access pattern mining algorithms.

Keywords: First Occurrence List Mine, web access pattern, **Web Access Database**

I.INTRODUCTION

Web usage mining, also known as Web log mining, discover interesting and frequent user access patterns from the web browsing details stored in server web logs, proxy server logs or browser. Web log mining has become very critical for effective Web Site Management, creating Adaptive Web Sites, business and support services, personalization, and network traffic flow analysis and so on. The compact WAP-Tree structure and the corresponding WAP-Mine algorithm performed better than all other earlier apriori methods. Studies conducted by many other researchers brought forward significant improvements to the WAP-Mine method. All these WAP-Tree based algorithms require two database scans; one for finding out the frequent items and the other for creating the aggregate tree using the frequent sub-sequences. Moreover, in spite of its compactness, the size of node and complexity of building up the WAP-Tree are disadvantageous. CSB-Mine algorithm is a web access pattern mining method that does not use WAP-tree. It uses a database structure called Conditional Sequence Base (CSB) to store the frequent access sequences. Using this database, intermediate projection databases are generated. The performance analysis shows that CSB-Mine outperforms WAP-Mine algorithm especially when support becomes smaller. CSB-Mine algorithm is more scalable when the input database size becomes larger. Motivated from this, in the proposed FOL-Mine, we have adopted the idea of eliminating WAP-Tree. All WAP-Tree based mining algorithms except WAP-Mine employ prefix building to generate patterns recursively. They use various techniques to locate the First Occurrences of each symbol so as to build patterns. Efficiency of the process of finding out the first occurrences of each symbol becomes crucial in the mining process. In FOF-Mine the authors give the idea of collecting the first occurrences together through the concept of Forest of First Occurrences. This concept is used in the proposed algorithm to improve the efficiency. This avoids the reconstruction of projection databases and thus saves space and time considerably. In the proposed FOL-Mine, a linked list with header node, stores the first occurrences of each event in a very compact and efficient way. Header node stores the count of first occurrences which gives the total support. The proposed algorithm avoids the unnecessary generations of intermediate projected databases and tedious support counting.

The concepts and algorithms which forms the basis of the proposed algorithm are discussed below.

Sequential Pattern Mining: Sequential Pattern is a sequence of item sets that frequently occurred in a specific order. Sequential Pattern Mining finds out the relationships between occurrences of sequential events, and also the specific order of the occurrences. Sequential Pattern Mining is used in a great spectrum of areas. In computational biology, sequential pattern mining is used to analyze the mutation patterns of different amino acids. Business organizations use sequential pattern mining to study customer behaviors. Sequential Pattern Mining is also used in system performance analysis and telecommunication network analysis.

Web Access Pattern Mining: As a web access sequence contains page view information of a user in a session in a time stamp order, sequential pattern mining can be adapted for mining the frequent access patterns in the set of web access sequences, WASD. Given a Web access sequence database WASD and a support threshold ξ, Web Access Pattern Mining mines the complete set of ξ-patterns of WASD.

WAP-Tree Based Mining of Web Access Pattern: proposed a compressed data structure known as Web Access Pattern Tree (WAP-tree) to hold web access sequences and WAP-Mine (Web Access Pattern Tree Mining) algorithm for mining web access patterns using the WAP-Tree. WAP-Tree facilitates support counting elegantly and enables the sharing of tree branch by common prefixes. It was proved to outperform all its earlier counterparts.

II.THE PROPOSED ALGORITHM

The proposed method involves three major components: *Main*, FOL-*Mine* and *Construct-FOL*. The *Main* algorithm reads in Web Access Sequences from the sequence database, prepares it for mining by loading it into the data structure and by setting the variables. *FOL-Mine* is the recursive mining algorithm used to mine all access patterns using the technique of suffix building. *Construct-FOL* is the algorithm used to build the first occurrence positions of the concerned frequent

item in the projected database.

Algorithm: Main

Main algorithm reads access sequences one by one from Sequence Database into the linked list and registers the start address in an array of pointers, Headlist. While scanning the access sequences, new symbols encountered are entered into the item list, the list of items, with count as 1, whereas for already existing symbols corresponding count is incremented. Node structure of the linked list where the web access sequences are stored is, *struct* node {symbol *item*; next **node* ;}. The start address of each linked list is stored in an array of pointers to structure node. When all sequences are read into the structure, list of frequent events are formed by selecting those items in the item list with count greater than the absolute support. Main algorithm then calls the recursive FOL-Mine function to generate the set of access patterns. Main algorithm to mine the complete set of Web Access Patterns that satisfy a predefined support threshold from Web Access Database, WASD

Algorithm Main

Input:

1. WASD, the Web Access Sequence Database.
2. Support=Support threshold.

Output:

F=the set of sequential access patterns

Method:

I. Read an access Sequence from file
II. Construct linked list
III. register the start address in *Headlist*[m]
IV. Update *itemset* and support
V. Increment *m*
End While

3. η =support**m*; // absolute support
4. Generate the set of frequent event, $\sum$
5. FOL-Mine (ε, *L*, η)
6. Return.

Algorithm: FOL-Mine

The recursive mining algorithm *FOL-Mine* works as follows: - The algorithm scans the set of frequent item. Assume that ***a*** is the current symbol. The algorithm uses the function *Construct-FOL* to generate the list of first occurrence position (FOL) *L* of ***a*** in the database. During the first call, function *Construct-FOL* scans the database itself to find out the first occurrence positions of the current symbol in various Access Sequences.

In the subsequent calls it uses the First Occurrence List (FOL) generated in the previous call. The header of FOL contains the total number of occurrences, that is the support of ***a*** either in the database or in the projected databases as the case may be. If the support is greater than the absolute support, the symbol is attached to the previously generated pattern and the recursive procedure is carried out until no more possibilities of expansion exists. If the support is not satisfied, the recursive call is terminated. This process is continued with all frequent symbols and terminated when no more patterns to be generated.

The Recursive mining algorithm to build pattern by suffix building

Algorithm: *FOL-Mine*

Algorithm FOL-Mine (pattern q, FOL L, η)
for each ***a*** $\in \sum$ do
La← Construct-FOL (L, ***a***)
if support of ***a*** > η F←F U{ q .*a* }
F'=F U FOL-Mine(q .***a*** , La, η) end if
delete La end for
return

Algorithm Construct-FOL

FOL is a linked structure used to store the First Occurrence Positions of a symbol. It has a header node containing label and total number of occurrences. Other nodes contain the sequence number and position of each of its First Occurrence. The header of the list holds the support count of the current symbol in the data base. In the initial call the database is the original database itself, but in the subsequent calls it is the projected database. First step of the algorithm initializes the count to zero. When the call to the *Construct-FOL* is made for the first time, the input list L will be empty. If it is not empty, it means that First Occurrences of the current symbol in the projected database is to be found out. So the content of FOL acts as the starting positions of the search for first occurrences. At the end of the search the header node is updated with number of occurrence of the current symbol. The list *FOL* is returned to the calling function for further processing.

FOL- Mine (pattern q, FOL L, η)
FOL-Mine (q ***.a***, La, η) delete La return

Algorithm for constructing first occurrence list

Algorithm: *Construct-FOL* (L, ***a***)
// *Algorithm for generating the first occurrence list FOL* //
1. Count_a=0
2. La = create a header node with label ***a***
3. if L is not empty
// *FOL is made using FOL in the previous call*//
for each item in L
find the next occurrence of item ***a***
If (exists)
create a node and attach to La increment Count_a.
end for. else
for each web access sequence
find the first occurrence of item *a*. if (exists)
create a node and attach to La increment Count_a.
end for. End if.
4. update header node with Count_a
5. return La

III.PERFORMANCE EVALUATION

Frequent sequential pattern mining algorithms fall into two categories; apriori based and pattern growth based. Many

studies reveal that the pattern growth methods outperform apriori ones. As *FOL-Mine* is a pattern growth method, comparison with other pattern growth based algorithms is enough to prove its efficiency. WAP-Mine and other WAP-Tree based methods are the prominent proposals in pattern growth approach. In authors proved *PLWAP-Mine* performs better than the *WAP-Mine* method in The authors of *FLWAP-Mine* Claim the better performance of *FLWAP-Mine* over . *PLWAP-Mine* in their performance evaluation. Thus, it is clear that FOF-Mine performs better than all other algorithms. CSB-Mine is a proposal that adopts concepts of pattern growth but avoids the use of complicated tree structures. Authors of CSB-Mine claimed that the algorithm performs better than WAP-Mine at lower supports. But In *CSB-Mine* intermediate sub conditional sequence bases are generated in each recursive call. This takes up a significant amount of time and space. But, in *FOL-Mine* once the sequences are represented in a liked format, rest of the mining is done using it. No intermediate reconstruction is required.

CSB-Mine includes two more procedure to carry out the mining. During each call it has to build an event queue. Algorithm Single Sequence Test is used to verify whether all sequences in sub conditional sequence bases can be combined into a single sequence. If so, the mining of that sub conditional sequence bases will be terminated. This single sequence will be used to form a part of the final sequential access patterns. Otherwise, sub conditional sequence base is constructed and mining is done recursively. Now, to prove *FOL-Mine* to be a better performer, it is enough to compare it with FOF-Mine. Time and Memory are the two major concerns in sequential pattern mining. So, these two factors are considered for performance evaluation of the proposed algorithm.

Execution Time Comparison

The execution time of the two algorithms was thoroughly tested. Codes are added in the original implementation to measure the total execution time. Experiments were repeated by varying the support threshold. In all data sets *FOL-Mine* algorithm is shown to outperform *FOF-Mine* especially when the support value becomes smaller.Comparison of Execution time is conducted using the four datasets. Figure.1 and Figure 2 show the comparison of execution time of *FOL-Mine* and FOF-Mine on databases T10I4D100K and msnbc respectively. Results of the comparison are also given in Table 1 and Table 2.

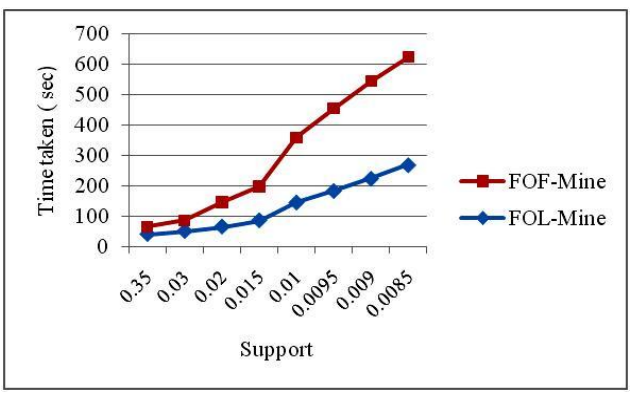

Figure 1: Execution Time Trend at Various Minimum Supports for T10I4D100K

Support	Execution time (sec) *FOL-Mine*	*FOF-Mine*	Number of Pattern
0.35	41	25	255
0.03	51	36	309
0.02	66	80	536
0.015	87	112	882
0.01	146	214	3300
0.0095	184	272	6893
0.009	225	321	11164
0.0085	270	355	13370

Table 1: Execution Time Trend and Number of Patterns at Various Minimum Supports for T40I10D100K

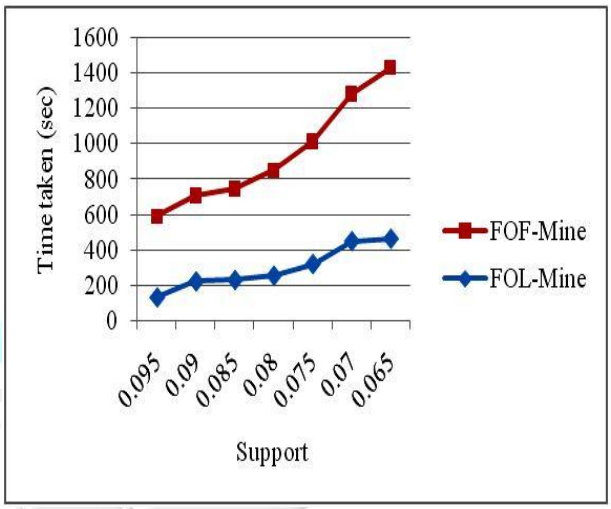

Figure.2: Execution Time Trend at Various Minimum Supports for T40I10D100K

support	Execution Time (sec) *FOL-Mine*	*FOF-Mine*	Number of patterns
0.095	134	460	92
0.09	224	485	110
0.085	232	517	120
0.08	257	596	137
0.075	322	695	157
0.07	451	831	183
0.065	465	965	208

Table 2: Execution Time Trend and Number of Patterns at Various Supports for T40I10D100K

IV.CONCLUSION

A new pattern growth, web access pattern mining algorithm FOL-Mine, is proposed in this paper. The proposed method FOL-Mine needs only one data base scan. In the single scan it

loads all access sequences in the linked structure and collects all necessary information for support counting. Instead of deleting the infrequent elements from the access sequences as in the earlier WAP-Tree based methods, the proposed method skips them efficiently during pattern mining. This prompts the use of the proposed linked database structure for incremental mining too. The proposed FOL-Mine avoids the use the complicated node structures. It uses much simple FOL structure to facilitate efficient mining. Correctness of the method is verified using test database. The efficiency of the method is evaluated by comparing the proposed method with earlier method both in terms of speed and memory. Experiments are done on both standard synthetic data sets and real time data sets. The new method outperforms the earlier method significantly. The scale-up experiments exhibits a linear increase in the execution time as the size of data base increases.

V.REFERENCE

[1] Agrawal, Imielinski and Swami, 1993] Agrawal, R., Imielinski, T and Swami, A. Mining Association Rules Between Sets of Items in Large Databases, Proceedings of 1993 ACM SIGMOD International Conference on Management of Data, vol. 22(2), pp. 207-216, 1993.

[2] Alonso., Grottke,M., Nikora,A.P, and Trivedi.KS., "An empirical investigation of fault repairs and mitigations in space mission system software", In Proc. 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2013.

[3] An Efficient Algorithm for Web Access Pattern Mining Fuzzy Systems and Knowledge Discovery, Fourth International Conference on (2007) Haikou, Hainan, China ISBN: 0-7695-2874-0

[4] Multimedia Data Mining: A Systematic Introduction to Concepts and Theory by Zhongfei Zhang, Ruofei Zhang