ISSN:0975-9646

Binate Code WAP Approach to access Sequential Pattern Mining for Web Log

[1]**Ms. Abhilasha Vyas**
M. Tech Scholar, PCST Indore

[2]**Ms. Priyanka Dhasal**
Assistant Professor, PCST Indore

Abstract:Data Explosion is the major challenge which our information industry is facing everyday. As World Wide Web has lots of data there is a need for discovery & analysis of useful information over the web. For this, we use web access pattern which is sequence of accesses followed by users frequently. Finding Web access pattern comes under sequential pattern mining which is the process of applying data mining technique to sequential database in order to discover co-relation relationship that exist among ordered list of events. Web access pattern tree (WAP) can be used to analyze web log access sequences, which first store the original web access sequences database on a pre-fix tree then WAP tree algorithm mine the frequent sequences from the WAP-tree in recursive manner by reconstructing intermediate tree starting with suffix sequences & ends with prefix sequences . We have an attempt to improve the efficiency of WAP tree approach. In Binate code WAP , We totally eliminate the need to reconstruct number of intermediate WAP-tree so that we can reduces execution time considerably.

Keywords: WAP tree, data mining, sequential data mining, frequent pattern tree

1 INTRODUCTION

Data Mining is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. With the wide spread use of databases and the explosive growth in their sizes, organization are faced with the problem of information overload. The problem of effectively utilizing these massive volumes of data is becoming a major problem for all enterprises.

Traditionally, we have been using data for querying a reliable databases repository via some well-circumscribed application for canned report-generating utility. While this mode of interaction is satisfactory for a large class of applications, there exist many other applications which demand exploratory data analyses. These applications support query-triggered usage of data, in the sense that the analysis is based on a query posed by a human analyst. On the other hand, data mining techniques support automatic exploration of data. Data mining attempts to source out patterns and trends in the data and infers rules from these patterns. With these rules the user will be able to support, review and examine decisions in some related business or scientific area. This opens up the possibility of a new way of interacting with databases and data warehouses.

Sequential mining is the process of applying data mining techniques to a sequential database for the purposes of discovering the correlation relationships that exist among an ordered list of events.

The objective of this work is to apply data mining techniques to a sequential database for the purposes of discovering the correlation relationships that exist among an ordered list of events. Given a WASD (Web Access Sequence Database),the problem to find frequently occurring Sequential patterns on the basis of minimum support provided. The application of sequential pattern mining are in areas like Medical treatment, science & engineering processes, telephone calling patterns. Sequential pattern mining Web usage mining for automatic discovery of user access patterns from web servers. It is used by an e-commerce company, this means detecting future customers likely to make a large number of purchases, or predicting which online visitors will click on what commercials or banners based on observation of prior visitors who have behaved either positively or negatively to the advertisement banners.

2. BACKGROUND

Sequential Pattern Mining comes in Association rule mining. For a given transaction database T, an association rule is an expression of the form X Y, where X and Y are subsets of A and X Y holds with confidence , if % of transactions in D that support X also Y. The rule X Y has support in the transaction set T if % of transactions in T support X U Y. Association rule mining can be divided into two steps. Firstly, frequent patterns with respect to support threshold min sup are mined. Secondly association rules are generated with respect to confidence threshold minimum confidence. Pattern Mining is of two types:

[1] **Non Sequential Pattern Mining:** The items occurring in one transaction have no order.

[2] **Sequential Pattern Mining:** The items occurring in one transaction have an order between the items (events) and an item may re-occur in the same sequence.

WAP-tree, which stands for web access pattern tree. The main steps involved in this technique are summarized next. The WAP-tree stores the web log data in a prefix tree format similar to the frequent pattern tree (FP-tree) for non-sequential data. The algorithm first scans the web log once to find all frequent individual events. Secondly, it scans the web log again to construct a WAP-tree over the set of frequent individual events of each transaction. Thirdly, it finds the conditional suffix patterns. In the fourth step, it constructs the intermediate conditional WAP-tree using the pattern found in previous step. Finally, it goes back to repeat Steps 3 and 4 until the constructed conditional WAP-tree has only one branch or is empty.

TID	Web Access sequence	Frequent Subsequence
100	pqspr	pqpr
200	tptqrp	pqrp
300	opqupt	qpqp
400	puqprur	pqprr

Table 1. Sequence database for WAP-tree

Thus, with the WAP-tree algorithm, finding all frequent events in the web log entails constructing the WAP-tree and mining the access patterns from the WAP tree. The web log access sequence database in Table 1 is used to show how to construct the WAP-tree and do WAP-tree mining. Suppose the minimum support threshold is set at 75%, which means an access sequence, *s* should have a count of 3 out of 4 records in our example, to be considered frequent. Constructing WAP-tree, entails first scanning database once, to obtain events that are frequent. When constructing the WAP-tree, the non-frequent part of every sequence is discarded. Only the frequent sub-sequences are used as input. For example, in Table 1, the list of all events is p, q, r, s, t, u and the support of p is 4, q is 4, r is 3, s is 1, t is 2, and u is 2.With the minimum support of 3, only p, q, r are frequent events. Thus, all non-frequent events (like *s, t, u*) are deleted from each transaction sequence to obtain the frequent subsequence shown in column 3 of Table 1.

With the frequent sequence in each transaction, the WAP-tree algorithm first stores the frequent items as header nodes so that these header nodes will be used to link all nodes of their type in the WAP-tree in the order the nodes are inserted. When constructing the WAP tree, a virtual root (Root) is first inserted. Then, each frequent sequence in the transaction is used to construct a branch from the Root to a leaf node of the tree. Each event in a sequence is inserted as a node with count 1 from Root if that node type does not yet exist, but the count of the node is increased by 1 if the node type already exists. Also, the head link for the inserted event is connected (in broken lines) to the newly inserted node from the last node of its type that was inserted or from the header node of its type if it is the very first node of that event type inserted. For example, as shown in figure 1(a), to insert the first frequent sequence pqpr of transaction ID 100 of the example database, since there is no node labeled p yet, which is a direct child of the Root, a left child of Root is created, with label p and count 1. Then, the header link node for frequent event p is connected (in broken lines) to this inserted *a* node from the p header node. The next event q is inserted as the left child of node p with a count of 1 and linked to header node q, the third event p is inserted as the left child of the node q having a count of 1, and the p link is connected to this node from the inserted p. The fourth and last event of this sequence is r and it is inserted as the left child of the second p on this branch with a count of 1 and a connection to r header node. Secondly, insert the sequence pqrp of the next transaction with ID 200, starting from the virtual Root (figure 1(b)). Since the root has a child labeled p, the node p's count is increased by 1 to obtain (p: 2). similarly, (q: 2) is also in the tree. The next event, r, does not match the next existing node p, and new node r:1 is created and Inserted as another child of q node. The third sequence qpqp of ID 300 and the fourth sequence pqprr are inserted next to obtain figure 1(c) and (d) respectively.

Once the sequential data is stored on the complete WAP-tree (figure 1(d)), the tree is mined for frequent patterns starting with the lowest frequent event in the header list, in our example, starting from frequent event r as the following discussion shows. From the WAP-tree of figure 1(d), it first computes prefix sequence of the base r or the conditional sequence base of *c as*: pqp:2; pq:1; pqpr:1; pqp:-1.

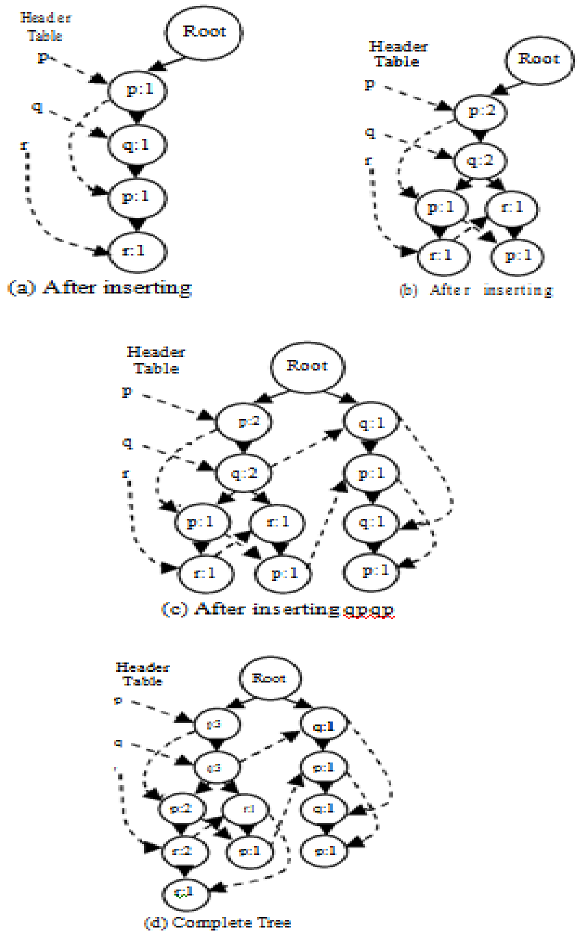

Figure 1. Construction of WAP Tree

WAP-tree|r, is built using the same method as shown in figure 1. The new conditional WAP-tree is shown in figure 2(a). Recursively, based on the WAP-tree in figure 2(a), the next conditional sequence base for the next suffix subsequence, qr is found as p(3). With p as the only frequent pattern in this base, the frequent sequence base of qr used to construct the next WAP tree shown in figure 2(b) is p(3). This ends the re-construction of WAP trees that progressed as suffix sequences |r, |qr and the frequent patterns found along this line are r, qr and pqr. The recursion continues with the suffix path |r, |pr. Thus, the conditional sequence base for suffix pr is computed from figure 2(a) as Ø, pq:3. This list is used to construct the WAP tree of figure 2(c). The algorithm keeps running, finding the conditional sequence bases of qpr as p: 3. from the list, the conditional frequent events of pqr is only p: 3. Then, the conditional WAP-tree|qpr is built as shown in figure 2(d). Now back to completing the mining of frequent patterns with suffix pr, figure 2(c) is mined for conditional sequence bases for suffix ppr and we get NULL.

The conditional search of r is now finished. The search for frequent patterns that have the suffix of other header frequent events (starting with suffix base |q and then |p) are also mined the same way the mining for patterns with suffix r is done above. After mining the whole tree, discovered frequent pattern set is: {r, qpr, pqpr, pr, pqr, qr, qb, pq, p, pp, qp, pqp}.

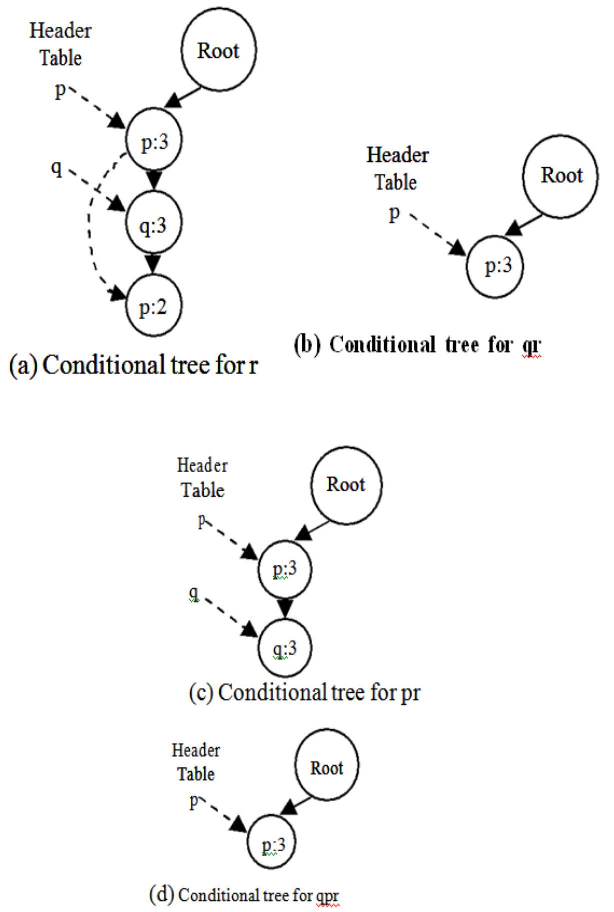

Figure 2. Reconstruction of WAP trees for mining conditional pattern base r.

3. RELATED WORK

3.1 Binary Coded Web Access Pattern tree Approach)

The tree data structure, similar to WAP-tree, is used to store access sequences in the database, and the corresponding counts of frequent events compactly, so that the tedious support counting is avoided during mining. A Binary code is assigned to each node in modified WAP-tree. These codes are used during mining for identifying the position of the nodes in the tree. The header table is constructed by linking the nodes in sequential events fashion. Here the linking is used to keep track of nodes with the same label for traversing prefix sequences. This mining algorithm is prefix sequence search rather than suffix search.

3.2 The Algorithm

Input : Access sequence database D(i), min support MS (0< MS ≤ 1)

Output : frequent sequential patterns in D(i).

Variables : Cn stores total number of events in suffix trees, A stores whether a node is ancestor in queue.

Begin
Scan D(i) to discover frequent individual events L;
Scan D(i) again .Create a root node of Tree T.
code(root)= NULL;
count = 0; {
For (each access sequence, fs in D(i)) {
Extract frequent subsequence F=($fs_1 fs_2$ . . . fs_n) by removing all events that are not in L;
current node -> leftmost_Child(root);
for (k=1 to n) {
if (current node = NULL)
{Create a new child node with position code equal to “1” appended
To position code of parent of current node ;}
elseif (current node = fs_k) { NdFd = true ;}
else { make current node point to current node sibling}
}
if (NdFd = true)
{count (fsk) ++;
Make current node point to fs_k ;}
Else {create new child node with position code of current node with
“0” appended at the end;
Make current node point to new created node ;} } }
From root node, do a sequential Traversal of Tree T to make appropriate linkage queue;
PATTERN_DIS (Suffix tree roots STR, Frequent sequence FS);
end;
PATTERN_DIS(R, F) {
If (STR=empty) return;
for (each suffix tree of event in L) {
Save first event in e_i queue to A;
if (event e_i is descendent of any event in STR, and is not descendent of A)
{Insert ei suffix tree header set STR’;
Add count of e_i to Cn;
Replace the A with e_i. ; }
If(Cn > MS)
{Append ei after FS to FS’;
print (FS’);
PATTERN_DIS (STR’,FS’); }

4 Experimental Results

This experiment uses fixed size database and different minimum support .The datasets and algorithms are tested with minimum supports between 0.8% and 10% against the 60 thousand (60 K) database.

From Table 2 and figure 7, it can be seen that

	time in secs at different supports				
Algorithms	2	3	4	5	10
WAP	745	505	325	285	145
Modified WAP	225	150	100	94	47

Table 2. Execution times for dataset at different minimum supports.

The execution time of every algorithm decreases as the minimum support increases. This is because when the minimum support increases, the number of candidate sequence decreases. Thus, the algorithms need less time to find the frequent sequences. The modified WAP algorithm always uses less runtime than the WAP algorithm. WAP tree mining incurs higher storage cost (memory or I/O). Even in memory only systems, the cost of storing intermediated trees adds appreciably to the overall execution time of the program. It is however, more realistic to assume that such techniques are run in regular systems available in many environments, which are not memory only,but could be multiple processor systems sharing memories and CPU's with virtual memory support.

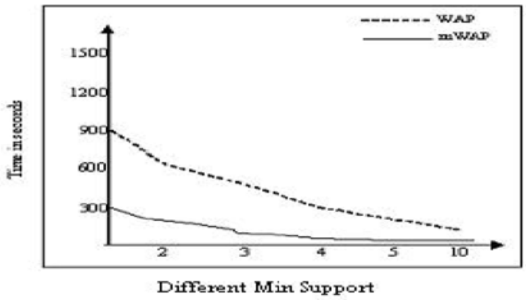

Figure 3. Execution times trend with different minimum supports.

Now, databases with different sizes from 20 K to 100 K with the fixed minimum support of 7% are used.

	Different changed transaction size				
Algorithms time in sec	20k	40k	60k	80k	100k
WAP	146	264	310	440	535
Modified WAP	45	72	95	141	175

Table 3. Execution times trend with different data sizes.

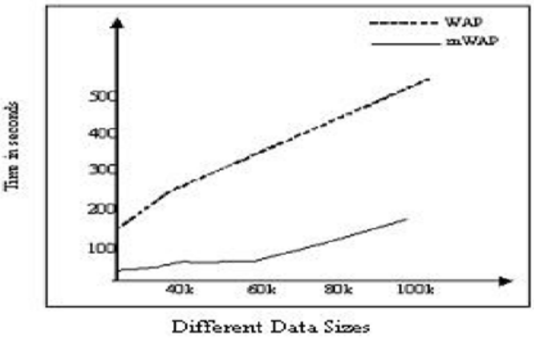

Figure 4. Execution times trend with different data sizes.

5 Conclusion

In this paper, we analyze the problem of sequential pattern mining. Here after discussing the two approached it is clear that the modified version is more efficient than the web access pattern tree approach. This presents a discussion of the advantages and disadvantages of both approaches conducted by comparing the performance with help of graph.

The modified algorithm eliminates the need to store numerous intermediate WAP trees during mining. Since only the original tree is stored, it drastically cuts off huge memory access costs, which may include disk I/O cost in a virtual memory environment, especially when mining very long sequences with millions of records. This algorithm also eliminates the need to store and scan intermediate conditional pattern bases for re-constructing intermediate WAP trees. This algorithm uses the pre-order linking of header nodes to store all events *ei* in the same suffix tree closely together in the linkage, making the search process more efficient. A simple technique for assigning position codes to nodes of any tree has also emerged, which can be used to decide the relationship between tree nodes without repetitive traversals.

References

[1] Agrawal, R. and Srikant, R. Mining sequential patterns. In Proc. 1995 Int. Conf. Data(ICDE'95), p.3–14, March 1995.

[2] Agrawal, R. and Srikant, R.,. Fast algorithms for mining association rules in large databases. In Proceedings of the 20th International Conference on very Large Databases Santiago, Chile, p.487–499, 1994.

[3] A. Nanopoulos and Y. Manolopoulos. Mining patterns from graph traversals. Data and Knowledge Engineering, 37(3):243–266, 2001.

[4] Etzioni, O. The world wide web: Quagmire or gold mine. Communications of the ACM, p.65 – 68, 1996.

[5] Han, J., Pei, J. et al. FreeSpan: Frequent pattern projected sequential pattern mining. In SIGKDD, p.355–359, Aug. 2000.