MSApriori using Total Support Tree Data Structure

Devashree Rai
Department of Electrical Engineering
National Institute of Technology, Raipur
Chhattisgarh, India

Kesari Verma
Department of Computer Application
National Institute of Technology, Raipur
Chhattisgarh, India

A.S. Thoke
Department of Electrical Engineering
National Institute of Technology, Raipur
Chhattisgarh, India

ABSTRACT

Association rule mining is one of the important problems of data mining. Single minimum support based approaches of association rule mining suffers from "rare item problem". An improved approach MSApriori uses multiple supports to generate association rules that consider rare item sets. Necessity to first identify the "large" set of items contained in the input dataset to generate association rules results in high storage and processing time requirement. The proposed work overcomes this drawback by storing items and their support values as total support tree data structure, resulting in an algorithm that is more efficient than existing algorithm both in terms of memory requirement as well as in processing time.

General Terms

MSApriori Algorithm, Total Support Tree Data Structure, Association Rule mining, Data Mining.

1. INTRODUCTION

Data mining [1] is extraction of interesting information or patterns from data in large databases. Association rule mining [2] [3] is important model in data mining. Association rule shows relationships among sets of items in a transaction database. Association rule discovery has been an active research area since its introduction in 1993[2].

Basic terminology about association rules is as follows: Let $I = \{i_1, i_2, .. i_m\}$ be a set of items and Let D be a set of transaction where any transaction $T \in D$ is a set of items such that $T \subseteq I$. An item set containing k number of items is called k-item set. A transaction, $T \in D$ contains a set of items or an item set, $X \subseteq I$, if $X \subseteq T$. An association rule is an expression of the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \phi$. The rule $X \Rightarrow Y$ holds in D with support, $\sup(X \Rightarrow Y) = P(X \cup Y)$ and $conf(X \Rightarrow Y) = P(Y \mid X)$ confidence. The objective of association rule mining is to discover all association rules that have support and confidence value greater than the user-specified minimum support (minsup) and minimum confidence (minconf) value.

Real world datasets are mostly non-uniform containing frequently as well as relatively rarely occurring entities. But most of the data mining approaches ignore rare entities and discover knowledge by considering only frequently occurring entities. Since rare entities has useful knowledge patterns [4] [5], research efforts are going on to find improved approaches that extracts rare knowledge patterns like rare association rules and rare class identification[4].

Various algorithms are proposed [2] [3] [6] [7] to mine association rules. Approaches that are single minimum support based like Apriori suffers from "rare item problem" dilemma [8]. Therefore, to extract frequent item sets involving rare items, an improved approach known as Multiple Support Apriori (MSApriori) has been proposed in [5]. This approach uses multiple supports instead of single. In the researches, efforts are being made to develop improved algorithms based on multiple supports [9] [10] [11]. In [5] frequent item sets involving rare items are obtained by assigning minimum item support (MIS) value to each item. Then item sets has to satisfy the lowest MIS value among the respective items. The rules generated are then pruned based on confidence value. However, for this pruning, the large set of item sets contained in the input data set need to be identified which in turn requires an effective storage structure.

In this paper, we proposed an approach that uses total support tree (T-Tree) data structure to implement MSApriori algorithm, T-Tree is an efficient data storage mechanism for item set proposed in [12]. By applying this storage structure, need to scan database each time for generating set of item sets is eliminated. This makes the rule generation process faster.

2. LITERATURE REVIEW

In this section we briefly discuss MSApriori algorithm for mining association rules and total support tree storage mechanism.

2.1 MSApriori Algorithm

MSApriori is an association rule mining algorithm proposed to extract frequent item sets involving rare items and to give better performance in comparison with approaches that employs single minimum support. Approaches that are based on single minsup may suffer from "rare item problem" dilemma. The "rare item problem" is as follows: if minsup is set to high value, frequent item sets involving rare items are missed and if set to low value, the number of frequent item sets explodes. To overcome this drawback MSApriori algorithm uses multiple support. Frequent item sets involving rare items are discovered by assigning minimum item support (MIS) value to each item. The MIS value is calculated through item support value S (i_j) and user specified (β) proportional value that can vary between 0 and 1. For every item $i_j \in I$, $MIS(i_j)$ is calculated as per equation 1.

$$MIS(i_j) = \beta \times S(i_j), \text{ if } \beta \times S(i_j) \succ LS \qquad (1)$$

$$= LS \quad \text{else}$$

Where, LS corresponds to user-specified least support value.

In MSApriori approach, item set containing rare item or combination of rare and frequent items has to satisfy relatively lower value than the item set containing only frequent items, to be included in frequent item set as MIS value is calculated as percentage of the item support value. Therefore MSApriori approach has better performance in comparison with single minimum support based approaches as it addresses the "rare item problem". But this approach has a drawback of scanning database each time to generate large item set and their support values which are then compared with MIS values. The working of MSApriori is illustrated as follows:

Table 1. Transaction Data Set.

TID	Items
1	Item1, item2
2	Item3,item1, item2
3	Item4, item1, item2
4	Item5, item1, item2
5	Item5, item2
6	Item6, item7
7	Item6, item7
8	Item8, item5
9	Item8, item5
10	Item8, item5

Table 1 shows the dataset of 10 transactions. The working of MSApriori algorithm with β=0.75 and LS=20% is depicted in figure 1. Items are generated after scanning the dataset and their respective MIS value is calculated using equation 1. Items whose support value is greater than their MIS value included in frequent 1-itemset. Each frequent 1-itemset is joined with each other and are included in frequent 2-itemset if their support is greater than or equal to lowest MIS value of items in it.

Example 1: The working of MSApriori for the transaction dataset shown in Table I is depicted in Figure 1.

{item4}[S: 10, MIS: 20]
{item3}[S: 10, MIS: 20]
{item6}[S: 20, MIS: 20]
{item7}[S: 20, MIS: 20]
{item8}[S: 40, MIS: 30]
{item1}[S: 40, MIS: 30]
{item5}[S: 50, MIS: 40]
{item2}[S: 50, MIS: 40]

$\Rightarrow$

{item6, item7}
{item6, item8}
{item6, item1}
{item6, item5}
{item6, item2}
{item7, item8}
{item7, item1}
{item7, item5}
{item7, item2}
{item8, item1}
{item8, item5}
{item8, item2}
{item1, item5}
{item1, item2}
{item5, item2}

Fig 1: Working of MSApriori algorithm with β=0.75 and LS=20%.

Support value is assigned to each 8 items in above example after scanning the data set and their corresponding MIS value is calculated by equation 1. Items whose support value is less than MIS value are not included further, for example item3 and item4 are not considered as candidates to be included in frequent 2-itemset. In the same way algorithm continues to generate all frequent k-item sets based on multiple minimum support based criteria.

2.2 Total Support Tree (T-tree)

In Association rule mining, performance of algorithms depends on size of input data set as the number of possible combinations represented by the items scales exponentially with the size of records resulting in an increase in memory and time requirement of algorithms. There are various methods proposed to optimize the performance of algorithms. One way of doing that is to make use of downward closure property of item sets which says that that for a frequent item set, all its subsets are also frequent and thus for an infrequent item set, all its supersets must also be infrequent. This property can be used to avoid generation and support computation for all combinations. But still there is need to scan data set number of times.

The algorithm Apriori [2] makes use of downward closure property to generate association rules. It can be implemented using structures such as set enumeration trees [13]. Set enumeration tree enumerates item sets in a best first manner. The top level of the tree will contain the support for 1-itemsets, the second level for 2-itemsets, and so on.

The T-tree (Total Support Tree) is a compressed set enumeration tree structure. In T-tree, any sub branch nodes that are at the same level are organized as 1D-array where array indexes represents column numbers. This gives us advantage of direct indexing with the use of column numbers by building reverse version of tree.

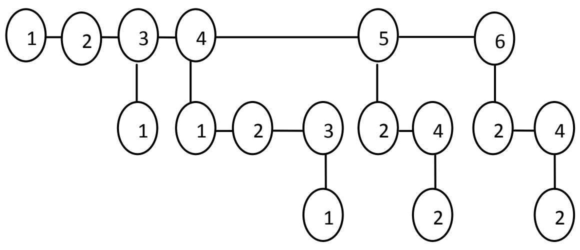

Fig.2. Total support tree structure for data set {1, 3, 4}, {2, 4, 5}, {2, 4, 6}

The T-tree uses index mechanism which makes traversal fast. In T-tree there is no need to explicitly store the item set labels and thus no sibling's reference variables are required. Total support tree structure for the data set r1:{1,3,4}, r2:{2,4,5}, r3:{2,4,6} is shown in figure 2, structure contains only combinations that actually appear in data set and thus eliminating need to generate each combinations. Level 1 represent single item sets and store support values for each singleton. Level 2 represents pair of two and soon.

3. PROPOSED WORK

To overcome the drawback of MSApriori algorithm that needs high storage requirement and processing time, we proposed an approach that combines the MSApriori algorithm with a total support tree storage structure resulting in a more efficient algorithm in terms of storage requirement and processing time. Implementation has structure of tree as shown in figure 5 with nodes at different level. Each total support tree node (TtreeNode) will have support value (sup), minimum support value (MIS) and reference to child nodes array (chdref). Tree is created level by level. Each level is created and corresponding support and MIS value is stored in it. The proposed algorithm (MSApriori-T) is as follows:

```
Procedure CreateMSApriori-T(){
1) CreateTtreeTopLevel ();
```

```
2) Prune (start, 1);
3) GenLevelN (start, 1, 1, null);
4) k=2;
5) While (isnewlevel)
6) Addsupport (k);
7) Prune (start, k);
8) isnewlevel = false;
9) GenLevelN (start, 1, k, {});
10) k++;
11) End
}
```

In the algorithm *start* is a reference to the start of the top-level array, N is the number of attribute in the data set, k is a level in T-tree, *and isnewlevel* is a Boolean variable initialized to false. Main function is CreateMSApriori-T () which calls other functions to build MSApriori-T tree structure. CreateTtreeTopLevel () creates top level of tree and add support (assuming count as support value) and MIS value to each node. Number of nodes created in first level will be equal to number of attributes N in the data set.

```
**CreateTtreeTopLevel ()**{
//create top level of Ttree
start[i] = new TtreeNode ();
//where 0< i < N
//Support and MIS value is added to nodes
start [$s_j$].sup++;
start [$s_j$].MIS = beta * start[$s_j$].sup;
if (start [$s_j$].MIS < LS)
start [$s_j$].MIS = LS;
}
```

The method TtreeNode is a constructor to build a new TtreeNode object. After the level is created prune() is called to prune the nodes which does not satisfy minimum support criteria.

```
**Prune(ref,k)**{
If (k=1) //level =1
    If (ref[t]! = null & ref[t].sup < ref[t].MIS)
    ref[t] = null; //nodes pruned
else
//other levels
    if (ref[t] != null & ref[t].chdref !=null)
    Prune (ref[t].chdref, k-1);
}
```

Similarly, other levels are generated by GenLevelN (). Their corresponding support and MIS value is added by Addsupport () function. Then nodes are pruned according to condition given in algorithm. Pseudo code for other methods is as follows:

```
**AddSupport (k)**{
    Addsup (start,k,|$r_i$|,$r_i$);
}
**Addsup (ref, k, end, r)**{
if (k==1)
    if (ref[$s_i$] != null)
        ref [$s_i$].sup++;
        ref [$s_i$].MIS= ref[$s_i$].sup * beta;
        if (ref[$s_j$].MIS  < LS){
         ref [$s_j$].MIS= LS;
        }
        If (parentof ref[$s_i$].MIS < ref[$s_i$].MIS){
        ref [$s_i$].MIS =    parentofref[$s_i$].MIS
        }
Else
If (ref[$s_i$] != null)
     Addsup (ref[$s_i$].chdref,k-1,I,r);
}
**GenLevelN** (ref, k, newk ,I){
if (k= newK)
    if(ref[i] != null ) genLevel(ref,I,append({i},I);
else
if( ref[i] != null)
    GenLevelN (ref[i].chdref, k-1, newK, append({i},I));
}
**GenLevelN (ref, end, I)**{
// create new array of t-tree nodes
Ref [end].chdref = new TtreeNode[end];
//Initialise elements where appropriate
If (ref[i] != null)
    newI= append ({i}, I);
    if (testcombinations(newI))
        ref [end].chdref[i] = new TtreeNode ();
        isNewLevel = true;
    else
            ref [end].chdref[i]= null;
}
**Testcombinations (I)** {
If (|I| < 3) return (true);
 $I_1$= {I [1], I[0]}
 $I_2$=delN (I, 2);
 Return(combinations(null,0,2,$I_1$ $I_2$));
}
```

```
Combinations (I, start, end, I_1, I_2) {
If (end > | I_2|)
    Testset = append (I, I_1);
    Return (FindInTtree (testset));
Else
    Tempset= append (I_2[i], I);
    If (~combinations (tempSet, i+1, end+1, I_1, I_2 ))
    Return (false);
    Return (true);
}
```

Figure 5 Shows the structure of the tree that is built by the proposed method for the data set r1:{1,3,4}, r2:{2,4,5}, r3:{2,4,6}, taking β=0.75. Level 1 in tree contains nodes for each single attribute, where each node have corresponding support and MIS value calculated by Equation 1. This level is pruned by method prune(start,1) ,it prunes the node which does not satisfy minimum support criteria. The method genLevelN () creates other levels of the tree. Nodes in level 1 has link to level 2 nodes indicating frequent 2-itemset. Level 2 nodes store support and MIS value of this pair. In our example itemset {1, 3} has support value 1 and 0.75 as its MIS value.

4. OBSERVATION AND RESULTS

The proposed algorithm is implemented in java 1.7.0, i5 core processor, windows 7 operating system. Figures are obtained by experimenting with different real data sets taken from UCI repository [14]. Table 1 shows the memory requirement of both the algorithms in bytes and Table 2 shows the running time of both the algorithms in seconds for different real data sets.

Table 2. Memory required by algorithms for different datasets.

Datasets	MSApriori Memory(in bytes)	MSApriori-T Memory(in bytes)
Car	7578128	2973376
Ecoli	5612984	1982296
Iris	2994456	1651960
Heart	3895584	2312688
Horsecolic	3214456	2312736
Pima	7356080	2651224
Glass	6611736	1652024
Tictactoe	14087936	2643040

Table 3. Time required by algorithms for different datasets.

Datasets	MSApriori Time(in sec)	MSApriori-T Time(in sec)
Car	0.187	0.05
Ecoli	0.141	0.06
Iris	0.078	0.01
Heart	0.234	0.08
Horsecolic	0.203	0.05
Pima	0.297	0.08
Glass	0.078	0.02
Tictactoe	0.161	0.03

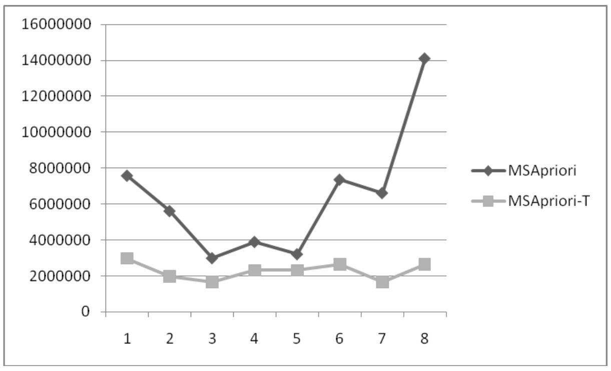

Fig 3. Comparison of Memory allocation between MSApriori and MSApriori-T algorithm

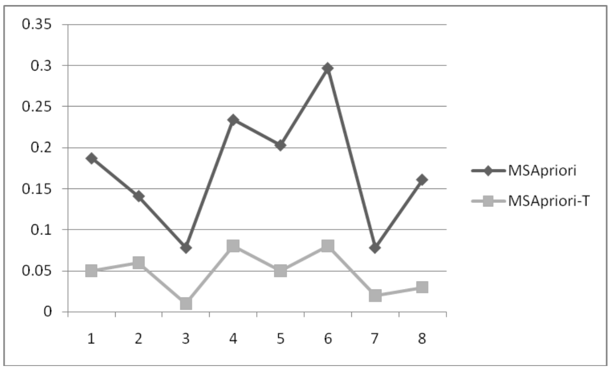

Fig 4. Comparison of running time between MSApriori and MSApriori-T algorithm.

4.1 Analysis

Above graphs (Figure 3 and Figure 4) shows the comparison of memory requirement (in bytes) and running time (in sec) of MSApriori and MSApriori-T algorithms. In our experiment we found that memory and time required by MSApriori-T algorithm is significantly reduced compared to MSApriori algorithm.

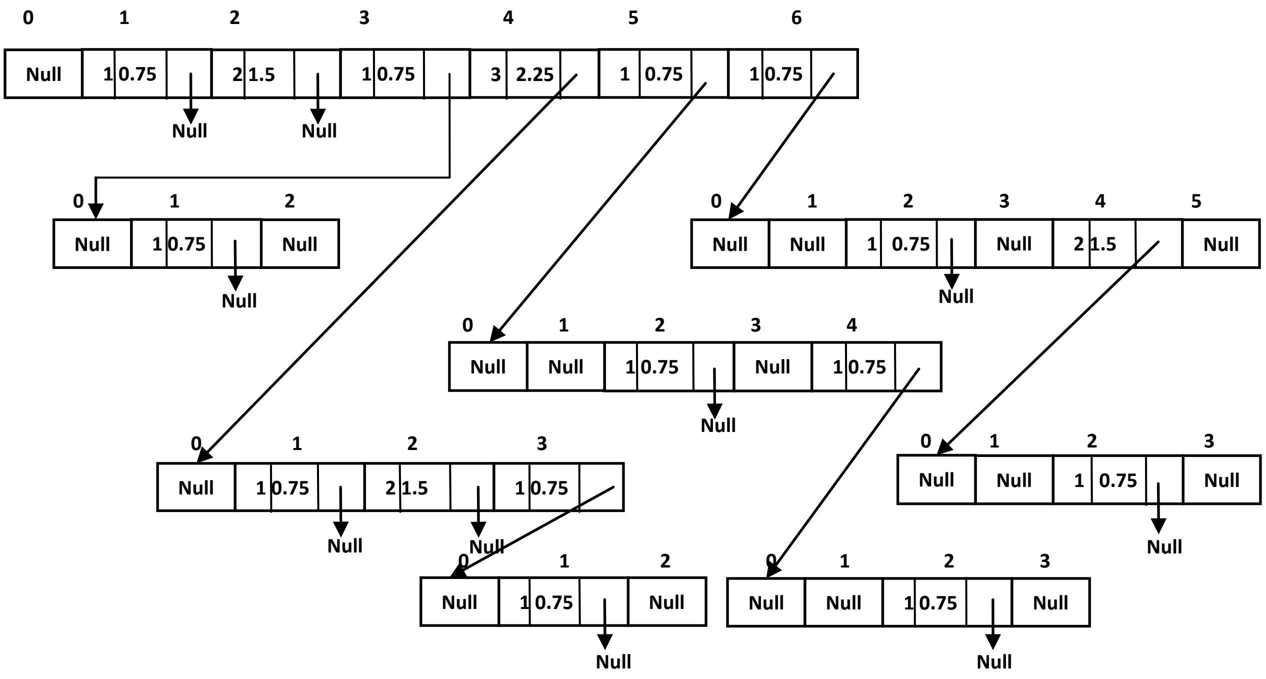

Fig 5: Tree structure for MSApriori-T algorithm

5. CONCLUSION

In Association rule mining MSApriori algorithm plays important role as it considers rare item sets. In this paper we proposed a novel approach MSApriori-T algorithm which uses total support tree structure to make MSApriori algorithm more efficient. T-tree stores each items in a tree as nodes and links are available to its child nodes. In the experiment it is found that proposed algorithm is faster and requires less memory comparatively. In future this approach could be used to enhance other association rule mining algorithms to make them more efficient.

6. REFERENCES

[1] M.S. Chen, J. Han, P.S. Yu, “Data mining: an overview from a database perspective”, *IEEE Transactions on Knowledge and Data Engineering*, 1996, 8, pp. 866-883.

[2] Agrawal, R., Imielinski, T., and Swami, A. “Mining association rules between sets of items in large databases.” SIGMOD, 1993, pp. 207-216.

[3] Agrawal, R., and Srikanth, R. “Fast algorithms for mining association rules.” VLDB, 1994.

[4] Weiss, G. M. “Mining With Rarity: A Unifying Framework.” SIGKDD Explorations, 2004, Vol. 6, Issue 1, pp. 7 – 19.

[5] Liu, B., Hsu, W., and Ma, Y. “Mining Association Rules with Multiple Minimum Supports.” SIGKDD Explorations, 1999.

[6] J.Han, Y. Fu, “Discovery of multiple-level association rules from large database”, in the twenty-first international conference on very large data bases, Zurich, Switzerland, 1995, pp. 420-431.W.-K. Chen, Linear Networks and Systems, Belmont, CA Wadsworth, 1993, pp. 123–135.

[7] Zaki, Mohammed J.; Scalable algorithms for association mining, IEEE Transactions on Knowledge and Data Engineering, vol. 12,no. 3, pp. 372-390, May/June 2000.

[8] Mannila, H. “Methods and Problems in Data Mining.” ICDT, 1997.

[9] R. Kiran and P. Reddy “An Improved Multiple Minimum Support Based Approach to Mine Rare Association Rules” IEEE 2009.

[10] I. Kouris, C. Makris, A. Tsakalidis ”An improved algorithm for mining association rules using multiple support values ” FLAIRS 2003.

[11] Lee, Hong and Lin “Mining association rules with multiple minimum supports using maximum constraints” Elsevier Science, Nov 2004.

[12] Coenen and Leng (2004). *Data Structures for Association Rule Mining: T-trees and P-trees* To appear in IEEE Transaction in Knowledge and Data Engineering.

[13] R. Rymon, “Search Through Systematic Set Enumeration,” Proc. Third Int’l Conf. Principles of Knowledge and Reasoning, pp. 539-550, 1992.

[14] Uci: Blake, c.l., & Merz, C.J (1998) UCI repository of machine leaning data bases from www.ics.uci.edu/~mlearn/MLrepository.html.