Improved algorithm for mining maximum frequent patterns based on FP-Tree

Naili Liu
Department of Information
Linyi University
Linyi,276005,China
lnl1999@163.com

Lei Ma
Department of Media
Linyi University
Linyi, 276005,China

***Abstract*—Mining association rule is an important matter in data mining, in which mining maximum frequent patterns is a key problem. Many of the previous algorithms mine maximum frequent patterns by producing candidate patterns firstly, then pruning. But the cost of producing candidate patterns is very high, especially when there exists long patterns. In this paper, the structure of a FP-tree is improved, we propose a fast algorithm based on FP-Tree for mining maximum frequent patterns, the algorithm does not produce maximum frequent candidate patterns and is more effectively than other improved algorithms. The new FP-Tree is a one-way tree and only retains pointers to point its father in each node, so at least one third of memory is saved. Experiment results show that the algorithm is efficient and saves memory space.**

Keywords- data mining;association rule;maximum frequent pattern;FP-Tree

I. INTRODUCTION

Mining frequent patterns in transaction databases has been studied popularly in data mining research, which reflects interesting association or contact between itemsets in the large amounts of data. Most of the previous studies adopt an Apriori[1,2] algorithm which adopts candidate set generation-and-test approach. However, candidate set generation is still costly, especially when there exists prolic patterns or long patterns[3,4]. J.Han put forward FP-Tree[5] algorithm which produces frequent itemsets by frequent patterns tree, which need not produce candidate itemsets and only need construct FP-Tree and conditional FP-Tree, which produces frequent patterns by visiting FP_Tree recursively. FP-Tree algorithm accesses database only twice. However, FP-Tree algorithm produces frequent patterns by mining conditional patterns and conditional FP-Tree recursively. Apriori algorithm and FP-Tree algorithm produce all frequent itemsets, but maximum frequent itemsets contain all frequent itemsets. So, mining frequent itemsets can be translated into mining maximum frequent itemsets.So,mining maximum frequent itemsets is very important in data mining.

Most of the previous studies of mining maximum frequent imtesets, such as Max-Miner[6],Pincer-Search[7], IMMFIA[8], DMFI[9] and DMFIA[10] and etc. Many improved algorithms were proposed based on these algorithms. Max-Miner algorithm adopts an Apriori-like approach,which is based on producing candidate itemsets and accessing database many times. DMFI algorithm adopts non-candidate itemsets, but still accesses database many times. DMFIA algorithm based on FP-Tree accesses database only twice, produces maximum frequent itemsets by non-conditional pattern and non-conditional FP-Tree, but still produces maximum candidate itemsets. This paper proposes an algorithm based on FP-Tree, which improves the structure of FP-Tree. Improved FP-Tree is a one-way tree and has only pointers to point its parent node and has no pointers to point its children in each node, so at least one third memory is saved, and has maximum non-candidate itemsets, therefore this algorithm improves the efficient of mining maximum frequent itemsets.

II. PROBLEM DESCRIPTION

A. Frequent patterns and maximum frequent patterns

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items, and a transaction database $D = \{T_1, T_2, \ldots, T_n\}$,where T_i $(1 \le i \le n)$ is a transaction which contains a set of items in I. For any $X \subseteq I$, we say that a transaction T contains X if there exists $X \subseteq T$. The set X is called an itemset. The support of an itemset X is the proportion of transactions in D which contain X, marked as X_{sup}. An itemset X is called frequent if its support is greater than or equal to given min_sup, where min_sup is called the minimum support.

Definition 1 Let X be an itemset, $X \subseteq I$, $X_{sup} \ge$ min_sup, and any of itemset Y, if there exists $X \subset Y$ and $Y_{sup} <$ min_sup, then X is a maximum frequent itemset in D.

B. Frequent Pattern Tree

Each node in the FP-Tree consists of five fields: node_name, node_count, node_fchild, node_link, and node_parent, where node_name registers which item this node represents, node_count registers the number of transactions represented by the portion of the path reaching this node, node_fchild registers the first child of this node, labeled as "null" if this node is leaf, which removes pointer of child node in ordinary FP-Tree , and node_link links to the next node in the FP-tree carrying the same item_name, or null if there is none. Each entry in the frequent-item header table in support ascending order consists of three fields:(1)item_name,(2)item_next and (3) head of item_head, which points to the minimum support node in the FP-tree carrying the item_name.

FP-Tree construction algorithm:

Input: A transaction database D and a minimum support threshold.

Output: frequent pattern tree, FP-tree

Method: The FP-tree is constructed in the following steps.

1. Scan the transaction database D once. Collect the set of frequent items F and their supports. We find out the list of frequent items L_{DF} by sorting F in support descending order.

2. Create the root of an FP-tree, and label it as "null".For each transaction in D do the following:

①Select and sort the frequent items in Trans according to the order of L_{DF}. Non-frequent items contained in the transaction without any treatment, because it will not be added to the FP-Tree. Let the sorted frequent item list in Trans be [p|P], where p is the first element and P is the remaining list. ②Call insert_tree([p|P],T).③If P is non empty, call insert_tree(P,N) recursively.The function insert_tree([p|P],T) is performed as follows:If T has a child N such that N.node_name=p, then increment N's count by 1;else create a new node N, and let its node_name be p and its node_count be 1,its node_parent link be linked to T, and its node_link be linked to the nodes with the same node_name via the node_link structure, and then make the following judgement: If T's node_fchild is empty, the newly created node N's node_name is assigned to the T's node_fchild in order to determine whether the node in the future use of leaf nodes, if node T's node_fchild is empty, it is the leaf node, otherwise it is not a leaf node.

III. MINING MAXIMUM FREQUENT ITEMSETS ALGORITHM

Lemma 1 If X is the maximum frequent itemset, then any subset of X is frequent itemset, and any maximum frequent itemset is a subset of L_{DF}.

This lemma is obtained according to the Apriori nature.

Lemma 2 If T's node_count is greater than or equal to minimum support min_sup, its predecessors nodes' node_count must be greater than or equal to min_sup.

Proof: According to the structure principle of frequent pattern tree FP-Tree, when frequent item in each transaction (each transaction has been ranked by support descending order) is added to the FP-Tree, the node count is increases by 1 if there exists the same name as the node, otherwise create a new node. Therefore, the count value of the parent node is greater than or equal the count value of the child node.

The basic idea of this algorithm: To find the maximum frequent itemsets according to item linked list of elements, linked list is established in support ascending order, so the first consideration is minimum support frequent itemset, each cycle do the following : identify the item you want to deal with the same name node in the FP-Tree nd_1,nd_2, ..., nd_h, then find out the path $p_1 \rightarrow p_2 \rightarrow \ldots \rightarrow p_h$ which cantains item according to node nd_1, nd_2, ..., nd_h in FP-Tree, and then process each node nd_1, nd_2, ...nd_h, firstly, judge p_i is a subset of the elements of MFS_D or not (that is, to determine whether the p_i is the element of MFS_D, or whether an element of MFS_D contains the p_i), if not, do the following: ①If node nd_i is a leaf node, and its support is greater than and equal to minimum support min_sup, then p_i is added to the MFS_D; otherwise whether there exists the same name node nd_j, satisfies $i \neq j$,and nd_i.node_count + nd_j.node_count≥min_sup and p_j contains p_i (that is, p_i is a subset of p_j) ,then add p_i to MFS_D; ② If the node nd_i is not a leaf node and its support is greater than or equal to min_sup, then add p_i to MFS_D; whether there exists the same name node nd_j, satisfies $i \neq j$,nd_i.node_count + nd_j.node_count≥min_sup and p_j contains p_i,then add p_i to MFS_D;

Mining maximum frequent itemsets algorithm:

Input:Frequent patterns tree FP-Tree;Frequent item link table Htable; minimum support threshold min_sup;Frequent item list L_{DF}

Output:Maximum frequent itemsets MFS_D

Method:

```
MFS_D=∅;
p=Head;
while (p!=null)
{
    find out nd_1,nd_2,…,nd_h nodes with the same name
        as p.item-name in the FP-Tree;
    find out the path p_1→p_2→…→p_h which contains
        p.item_name according to nd_1,nd_2,…,nd_h and
        pointer domain of its prefix node's parent node
    for (i=1;i≤h;i++)
    {
      if (p_i is not subset of any element of MFS_D)
       {
        if (nd_i.node_fchild=null)
        {
          if (nd_i.node_count≥min_sup)
           {
             if p_i is not subset of any element of MFS_D
               MFS_D=MFS_D∪p_i;
           }
           else for (j=1;j≤h;j++)
           {
              if (i≠j) and (nd_i.node_count+
             nd_j.node_count≥min_sup)and(p_j contains p_i)
                   MFS_D=MFS_D∪p_i;
           }
       }
       else
       if (nd_i.node_count≥min_sup)
          MFS_D=MFS_D∪p_i;
       else for (j=1;j≤h;j++)
          if (i≠j) and (nd_i.node_count+
           nd_j.node_count≥min_sup) and (p_j contains p_i)
                MFS_D=MFS_D∪p_i;
    }
    p=p.item_next;
}
```

Example:Tansaction database D is shown in Figure 1,minimum support count is 2(that is,s=2/9=22%),FP-Tree is shown in Figure 2.

TID	Item
T100	a,b,e

T200	b,d
T300	b,c
T400	a,b,d
T500	a,c
T600	b,c
T700	a,c
T800	a,b,c,e
T900	a,b,c

Figure 1. transaction database D

Node storage structure is node_name, node_count, node_fchild, node_link, node_parent,each node of FP-Tree records node_name,node_count,node_fchild.

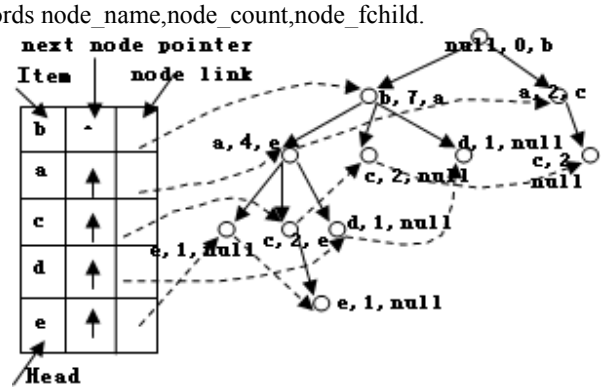

Figure 2. Htable and FP-Tree

First scan the database to come to frequent itemset F and support of frequent item, and produce frequent item list L_{DF} in support descending order F, and establish item linked list Htable in support ascending order of L_{DF}, in this example, e is first element in the list. The establishment of the FP-Tree tree is shown in Figure 2. According to the algorithm, assuming that when we analyze c in the project list, first find out three nodes with the same name as c in the FP-Tree according to node chain, respectively, and find the path containing c p_1 = {b, a, c,}, p_2 = {b, c}, p_3 = {a, c}, then analyze three nodes respectively, the first node nd_1=[c,2,e], because of nd_1.node_fchild = e, so this node is not a leaf node, and that p_1= {b, a, c} is not subset of some element of MFS_D = {{b,a,e}, {b, d}}, nd_1.node_count = 2≥min_sup, directly add p_1 = {b,a,c} to MFS_D, then MFS_D = {{b,a,e}, {b,d}, {b,a,c}}, then analyze node nd_2 = [c, 2, null], due to nd_2.node_fchild = null, so nd_2 is a leaf node, nd_2.node_count = 2≥min_sup,however, p_2 = {b, c}is subset of {b,a,c} in MFS_D, so p_2 can not be added to the MFS_D, then we continue to analyze node nd_3= [c,2,null], nd_3 is also a leaf node,nd_3.node_count = 2≥min_sup, because p_3 = {a, c} is also subset of {b,a,c} in MFS_D ,so p_3 can not be added to the MFS_D. This analysis of c in project linked list is over, and we continue to analyze other projects in the linked list until we analyze the end of the project linked list.

IV. ALGORITHM AND COMPARE

We use Visual Basic 6.0, memory 1G, CPU Celeron 2.5GHZ, Windows 2000 Server to realize this algorithm and DMFIA algorithm, this algorithm and DMFIA algorithm have the same number of times to scan database, and have the same time to establish FP-Tree because of only node structure different each other.The difference of two algorithms is execution time of finding maximum frequent itemsets. The best case time complexity of this algorithm is kh (where k is the number of frequent item, namely the number of frequent 1 - itemsets, h is the number of nodes in the FP-Tree and analysis frequently have the same item_name), the worst case time complexity of this algorithm is kh2,however, time complexity of DMFIA algorithm is a*k*(b*h+b*c) where a registers the |MFCSD| number in cycle, b registers the average of the number of |MFCSi| in cycle, and c registers the average of the number of per cycle MFCSi items,in the best case, DMFIA's time complexity is slightly larger than kh, in the worst case, its time complexity is about k2h2. From the above analysis can be drawn, this algorithm can be more effectively to find the maximum frequent itemsets than DMFIA algorithm. Experiment result is shown in Figure 3.

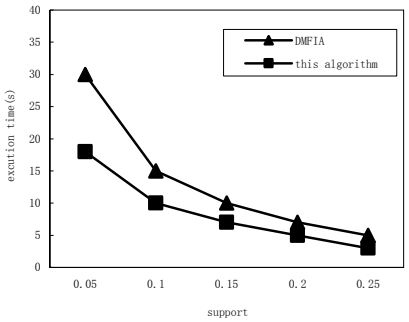

Figure 3. Experiment result

V. CONCLUSION

Finding maximum frequent patterns is one of the key issues in the field of data mining, it is also a hot research in data mining.In this paper an improved algorithm is presented, the algorithm does not generate candidate itemsets, and need not produce conditional FP-tree recursively, ant its time complexity is relatively low in mining maximum frequent patterns, thereby the efficiency of the algorithm is enhanced.

REFERENCES

[1] Agrawa lR, Imielinski T, Swami A. Mining association rules between sets of items in large databases (C). In: Buneman P, Jajodia S,eds. Proc. of the ACM SIGMOD Conf. on Management of Data (SIGMOD'93). New York: ACM Press, 1993. 207~216.

[2] Agrawa lR, Srikant R. Fast algorithms for mining association rules in large databases. In: Bocca JB, Jarke M, Zaniolo C, eds. Proc.of the 20th Int'l Conf. on Very Large Data Bases. Santiago: Morgan Kaufmann, 1994. 478~499.

[3] Aly HH, Taha Y, Amr AA. Fast mining of association rules in large-scale problems. In: Abdel-Wahab H, Jeffay K, eds. Proc. of the 6th IEEE Symp. on Computers and Communications (ISCC 2001). New York: IEEE Computer Society Press, 2001. 107~113.

[4] Tsai CF, Lin YC, Chen CP. A new fast algorithms for mining association rules in large databases. In: Kamel AE, Mellouli K, Borne P, eds. Proc. of the 2002 IEEE Int'l Conf. on Systems, Man and Cybernetics (SMC 2002). IEEE Computer Society Press, 2002.251~256.

[5] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation. In: Chen WD, Naughton J, Bernstein PA, eds. Proc. Of

the 2000 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2000). New York: ACM Press, 2000. 1~12.

[6] Bayardo R. Efficiently mining long patterns from databases. In: Haas LM, ed. Proceedings of the ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 1998. 85~93.

[7] Lin DI, Kedem ZM. Pincer-Search: A new algorithm for discovering the maximum frequent set. In: Schek HJ, ed. Proceedings of the 6th European Conference on Extending Database Technology. Heidelberg: Springer-Verlag, 1998. 105~119.

[8] Ma Li-sheng,Yao Guang-shun.Mining algorithm for maximal frequent itemsets based on improved FP-Tree.Journal of Computer Application,2012,32(2):326-329.

[9] Lu SF, Lu ZD. Fast mining maximum frequent itemsets. Journal of Software, 2001,12(2):293~297 (in Chinese with English abstract).

[10] Song Qingyu,Zhu Yuquan. An Algorithm and Its Updating Algorithm Based on FP-Tree for Mining Maximum Frequent Itemsets.Journal of Software, 2003,Vol14,No.9.