Volume 2, No. 9, September 2011

Journal of Global Research in Computer Science

RESEARCH PAPER

Available Online at www.jgrcs.info

ISSN-2229-371X

IMPLEMENTATION OF ARRAY BASED TECHNIQUE TO IMPROVISE REPRESENTATION OF FP-TREE USING IAFP-MAX ALGORITHM

Kuparala Chakrapani*[1]
[1]CSE Student
A.Lakshman Rao [2]
[2]Associate Professor
[1,2]M.Tech, Pragati Engineering College, Kakinada, India.
[1]chakrapani.kuparala@gmail.com
[2]laxman1216@gmail.com

Abstract: The essential aspect of mining association rules is to mine the frequent patterns. Due to intrinsic difficulty it is impossible to mine complete frequent patterns from a dense database. The quantity of mined patterns is generally large and it is firm to understand and utilize them. all frequent patterns are enclosed and compressed to maximal frequent patterns where the memory needed for storing them is smaller than that is required for storing complete patterns. Consequently, mining maximal frequent patterns provides a great value. This paper inorder to improve the structure of traditional FP-Tree presents an effective algorithm called IAFP-max for mining maximal frequent patterns based on improved FP-tree and array technique. The implementation of concept postfix sub- tree in the respective algorithm avoids generating the candidate of maximal frequent patterns in the mining process. Thus it reduces the memory consumed and also uses an array –based technique to the improved FP-Tree to reduce the traverse time. By the practical facts ,it represents that this algorithm overtakes many existing algorithms like MAFIA, Genax and FP max.

Keywords: FP-Tree, IAFP, Frequent Patterns, itemsets, Array Technique.

INTRODUCTION

Association rules is an important research field on KDD presented firstly by those people including Agrawal. Rules represent the interesting associations or relations between item sets in database. Mining frequent patterns is the fundamental and essential part in many data mining applications including the discovery of association rules, sequential patterns, and soon. Apriori algorithm and its improved ones are mostly adopted to mine frequent item sets in the past researches, all these algorithms make use of the character "all subsets of frequent pattern are frequent", but generate plenty of candidate item sets in mining process, and need to scan the original database many times, thus the mining efficiency is cut down.

Therefore, people including Jiawei Han presented an algorithm without candidate generation—FP-growth, which only need to scan database twice: frequent 1-item sets are gained in the first scanning database, and in the second time non-frequent item sets in original database are filtered with frequent 1-item sets and the FP-tree is built, then the complete frequent patterns are mined by performing recursively mining method in FP-tree. Experiments show that FP-growth is about an order of magnitude faster than Apriori. The drawback of mining complete frequent item sets is that if there is a large frequent item set with size *l*, then almost all 2*l* candidate subsets of the items might be generated. However, since frequent item sets are upward closed, it is sufficient to discover only all maximal frequent item sets. In addition, there are some mining applications which only need to discover maximal frequent patterns, not to discover complete ones, thus it is greatly significative to mine maximal frequent patterns.

The previously presented representative algorithms for mining maximal frequent patterns are MaxMiner, Depth Project, MAFIA, GenMax and FPmax, etc. MaxMiner extends Apriori to mine only "long" patterns (maximal frequent item sets). To reduce the search space, it performs not only subset infrequency pruning such that a candidate item set that has an infrequent subset will not be considered, but also a "look ahead" to do superset frequency pruning. Though superset frequency pruning reduces the search time dramatically, it still needs many passes to get all long patterns. Depth Project performs a mixed depth-first/breadth first traversal method and also use both subset infrequency pruning and superset frequency pruning. In the algorithm, the database is represented as a bitmap. Each row in the bitmap is a bit vector corresponding to a transaction, each column corresponding to an item. Experiments show that it outperforms Maxine by at least an order of magnitude.

MAFIA is similar to Depth Project, also uses a bitmap representation, where the count of an item set is based on the column in the bitmap (called "vertical bitmap"). To get the bit vectors for any item set, the bit vector *and*-operation need to be applied on the bit vectors of the items for any item set. Besides subset infrequency pruning and superset frequency pruning, the pruning technique called *Parent Equivalence Pruning* is also used in MAFIA. GenMax also uses a vertical representation of the database. However, for each item set, it stores a t*ransaction identifier*, or TIS, rather

than a bit vector. The algorithm takes a novel technique called *progressing focusing* to maximality testing. This technique maintains a set of local maximal frequent item sets, LMFI's. The newly found FI is firstly compared with item sets in LMFI. Most nonmaximal FI's can be detected by this step, thus reducing the number of subset tests.
FPmax extends the earlier algorithm FPmax, it only scans all FP-tee once by using array technique and uses MFI-tree to store all already discovered MFI's, adopts efficient method for subset testing.

Experimental evaluation shows that FPmax outperforms MAFIA and GenMax in many cases, especially for datasets with short average transaction length and long average pattern length. In this paper, an efficient algorithm, called IAFP-max, for mining maximal frequent patterns based on improved FP-tree and array technique is proposed; the algorithm improves the conventional FP-tree and by introducing the concept of the postfix sub-tree, avoids generating the maximal frequent candidate patterns in mining process and therefore greatly reduces the memory consume, it also uses an array-based technique to reduce the traverse time to the improved FP-tree. So it greatly improves the mining efficiency and saves the cost in time and space.

SYSTEM OVERVIEW

FP-tree and conditional FP-tree built by FP-growth algorithm need to generate in a top-down order, but the mining process of frequent patterns employs the bottom-up strategy. Because of the recursively generation of conditional FP-tree, both FP-tree and conditional FP-tree need to be able to traverse in two directions, the nodes in these trees require plenty of pointer, thus a great deal of memory is required for saving FPtree and conditional FP-tree. Improved FP-tree (IFP-tree) is similar with FP-tree and each node in IFP-tree consists of four fields: *item*, *count*, *ahead* and *next*. Where *item* registers which item this node represents, *count* registers the number of transactions represented by the portion of the path reaching this node, *ahead* links to the left child or the parent of the node, and *next* links to the right brother of the node or the next node in IFP-tree carrying the same *item*, or null if there is none. We also define two arrays:
nodecnt and *link*, and *link*[item] registers a pointer which points to the first node in the IFP-tree carrying this *item*,
nodecnt[item] registers the support count sum of those nodes in IFP-tree which carry the same *item*.
For example, let transaction database T be illustrated by TABLE I, and the minimum support (min_sup) be 4, then we can get the list (L) of frequent items, L = {(c,6)→(f,6)→(a,5) →(b,4)→(m,4)→(p,4)}, then IFP-tree is illustrated by Fig .

Table [1] : Transaction Database

TID	Item Set	S_t
t_1	a b c f m o	c f a b m
t_2	a c f q d	c f a
t_3	a f h m b	f a b m
t_4	b c k s p	c b p
t_5	f a c d i m p	c f a m p
t_6	f h j r	f
t_7	a f c e l p m n	c f a m p
t_8	d c g b k p	c b p

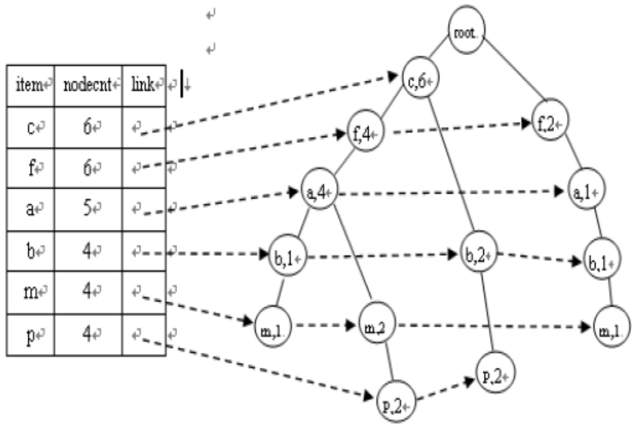

Figure 1. IFP-tree of transaction T.

Mining Maximal Patterns:

An Array Technique:

The main work done in the mining process is traversing the postfix sub-tree to count the support of item sets and constructing new postfix sub-tree, Recall that for each item i of conditional PT(x), two traversals of PT(x) are needed for constructing the new sub-tree PT(k,i). The first traversal finds all frequent items and their counts of support, the second traversal constructs the new sub-tree PT(k,i). In this paper, we use the array technique.

f	4				
a	4	5			
b	3	2	2		
m	3	4	4	2	
p	4	2	2	2	2
	c	f	a	b	m

(a) A_ϕ

f	2	
a	2	2
	c	f

(b) A(b)

Figure 2: Two Array Examples

The following example will explain the idea. In TABLE I, supposing that the minimum support is 4, after the first scan of the original database, we sort the frequent items as c:6,f:6,a:5,b;4,m:4,p:4. During the second scan of the database we will construct PT and an array A, this array will store the counts of all 2-itemsets. All cells in the array are initialized as 0.

The following is the main procedure of algorithm:

```
For each i ∈ L do
  {
   PT(i)= PT_construct ( IFP-tree, i);
   Mine LMFP_i from PT(i);
   MFPs = MFPs ∪ LMFP_i;
  }
}
```

In the step of MFPs ∪ $LMFP_i$, if $LMFP_i \subseteq MFPs$, then MFPs keep changeless, else insert $LMFP_i$ into MFPs, then performs *subset-checking*: if there is a pattern $I \subseteq MFPs$, and $I \subseteq LMFP_i$, then delete I from MFPs.

Let the items in L be ranked into $1 < \cdots < k$, and $1 \leq i \leq k$.

The following is the procedure of constructing PT(i):

```
Procedure PT_construct (IFP-tree, i )
 {
   for j = 1 to i-1 do
   link[j] = null and nodecnt[j] = 0;
   for each node N in link[i] do
   {
    basecnt = N.count;
      //basecnt is the base count
    let M be the parent of node N;
     while (M.item ≥ 1) do
       {
        if M isn't in link[M.item]
             Then insert M into link[M.item] and
             M.count = basecnt;
       else
             M.count = M.count + basecnt;
       nodecnt[M.item] = nodecntp[M.item] + basecnt;
        let M be its new parent again;
       }
     }
 }
```

By using the algorithm, the set of maximal frequent patterns is as follows: {{(c,f,a),4},{(b),4},{(f,a,m),4},{(c,p),4}}.

SYSTEM DESIGN & IMPLEMENTATION

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

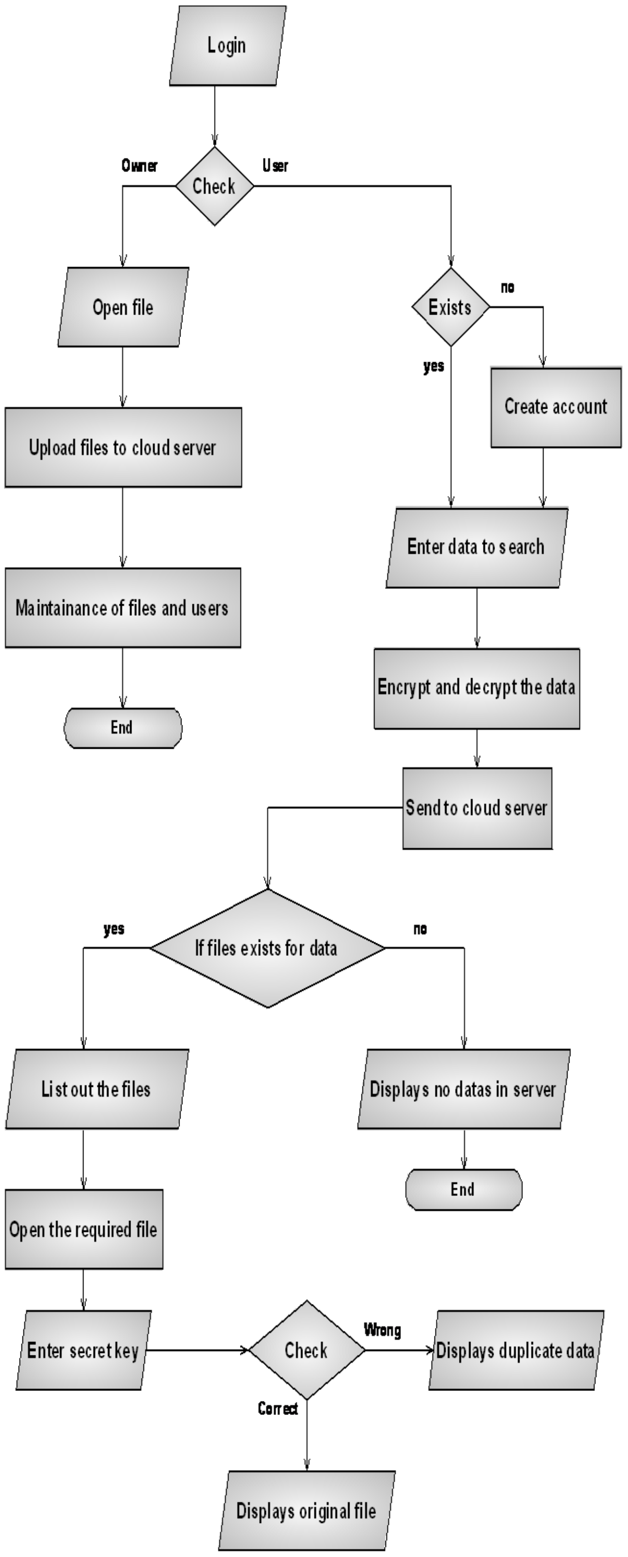

Figure 3: Inter-Operational DFD for the Framework

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

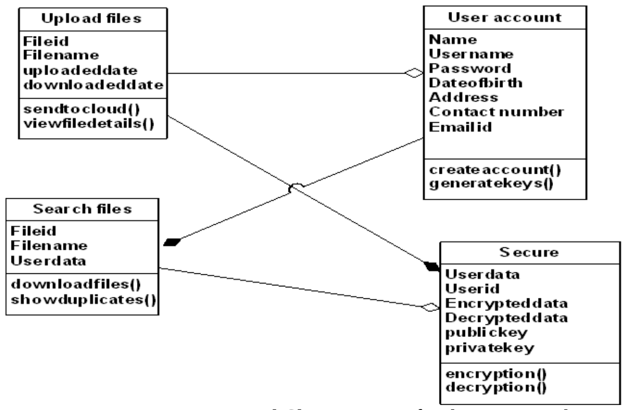

Figure 4: Inter-operational Class Diagram for the Framework

RESULTS

The experiments were conducted on 2.4 GHz Pentium IV with 512 MB of memory running Microsoft Windows XP. All codes was compiled using Microsoft Visual C++ 6.0. We used Connect-4 downloaded form a website to test and compared IAFP-max with GenMax, MAFIA and FPmax, which is a real and dense dataset. Fig 4 shows the experimental results. Here we can see that FPmax outperforms GenMax and MAFIA for high levels of minimum support, but it is slow for very low levels. On the other hand, IAFP-max outperforms FPmax and it is about one to two orders of magnitude faster than GenMax and MAFIA for all levels of minimum support.

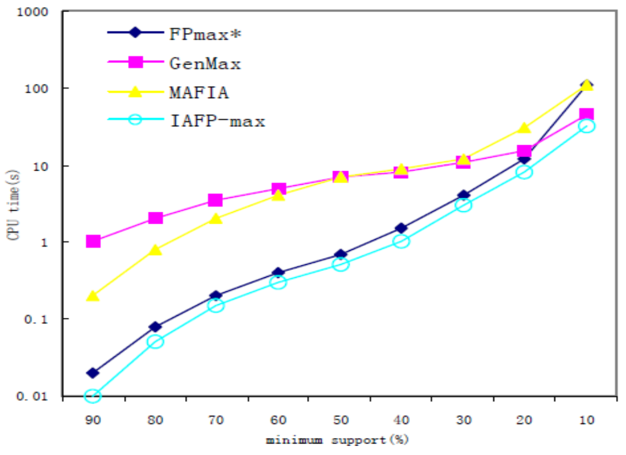

Figure 4. Comparison in Connect-4

CONCLUSIONS

In this paper, based on improved FP-tree and array technique, for mining maximal frequent patterns an efficient algorithm, called IAFP-max is proposed. The conventional FP-tree is improved by algorithm and by providing the concept of post fix sub tree. In mining process, generating the maximal frequent candidate patterns can be avoided by providing the concept of postfix subtree. This reduces the memory consumed and also uses an array based technique to the improved FP-Tree inorder to reduce the traverse time .Hence it highly increases the mining efficiency in time and space scalability. By the practical facts, it shows that it possess high mining efficiency and scalability.

REFERENCES

[1]. V. Anantharam, P. Varaiya, and J. Walrand, “Asymptotically Efficient Allocation Rules for the Multiarmed Bandit Problem with Multiple Plays—Part i: i.i.d. Rewards,” IEEE Trans. Automatic Control, vol. 32, no. 11, pp. 968-976, Nov. 1987.

[2]. R Agrawal, T Imielinski and A Swami, “Mining association rules between sets of items in large database’, Proceedings of the ACM SIGMOD International Conference Management of Date, Washington, pp. 207-216,1993..

[3]. A.L. Baraba´si and R. Albert,“Emergence of Scaling in Random Networks,” Science, vol. 286, pp. 509-512, 1999.

[4]. S. Brams and P. Fishburn, Approval Voting. Birkhauser, 1983.

[5]. J Han, J Pei and Y Yin, “Mining frequent patterns without candidate generation”, Proceedings of Special Interest Group on Management of Data, Dallas, pp. 1-12, May 2000.

[6]. R J Bayardo, “Efficiently mining long patterns from databases”, Proceedings of special Interest Group on Management of Data, Seattle, WA, pp . 85-93, June 1998.

[7]. S. Golder and B.A. Huberman, “The Structure of CollaborativeTagging Systems,” J. Information Science, vol. 32, no. 2, pp. 198-208,2006.

[8]. R C Aggarwal, C C Agarwal and V V V Prasad, “Depth First Generation of Long Patterns”, Proceedings of the 6th ACM SIGMOD International Conference on Knowledge Discovery & Data Mining, Boston, MA, USA, August 20-23, 2000

[9]. Junqiang Liu,Yunhe Pan"Mining frequent item sets by opportunistic projection",ISBN:1-58113-567-X doi>10.1145/775047.775081

[10]. Burdick Doug, Calimlim Manuel, and Gehrke Johannes, “A Maximal Frequent Itemset Algorithm for Transactional Database”, Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, pp . 443-452, April 2001.

[11]. S Goldwasser,S Micali,C Rackoff,"The knowledge complexity of interactive proof-systems",ISBN:0-89791-151-2 doi>10.1145/22145.22178.