# A Novel Approach for finding Frequent Item Sets with Hybrid Strategies

J.R.Jeba
Assistant Professor & Head
Department of Computer Applications
Noorul Islam Centre For Higher Education
Noorul Islam University, Kumaracoil

Dr.S.P.Victor
Associate Professor & Head
Department of Computer Science
St.Xavier's College (Autonomous)
PalayamKottai.

## ABSTRACT

Frequent item sets mining plays an important role in association rules mining. Over the years, a variety of algorithms for finding frequent item sets in very large transaction databases have been developed. Therefore, a number of methods have been proposed recently to discover approximate frequent item sets. This paper proposes an efficient SMine (Sorted Mine) Algorithm for finding frequent item sets. This proposed method reduces the number of scans in the database. Our proposed SMine algorithm works well based on graph construction. At last we performed an experiment on a real dataset to test the run time of our proposed algorithm. The experiment showed that it was efficient for mining datasets.

## General Terms

Data Mining, Frequent Item sets, Association Rule Mining.

## Keywords

SMine,item_count, frequent_items.

## 1. INTRODUCTION

Association rule mining is a focused area in today's data mining research. It usually consists of two phases viz., discovery of frequent itemsets and generation of rules from the discovered frequent itemsets. Finding frequent itemsets has gained popularity because it has more number of applications.

A number of algorithms for mining frequent item sets have been proposed after Agrawal first introducing the problem of deriving categorical association rule from transactional databases in [1]. These existing algorithms uses the candidate generate-and-test approach and the pattern growth approach. Apriori [2] and its several variations belong to the first approach, while FP-growth [7] and H-Mine [8] are examples of the second. In Apriori[1, 2] as well as many subsequent studies[3, 4], each iteration of the candidate generate-and-test approach, pairs of frequent k-item sets are joined to form candidate (k+1)-item sets, then scanned the database to verify their supports. The Apriori algorithm achieves good reduction on the size of candidate sets, however, it takes many scans of the database to check the candidate item supports as much as the most long length of patterns. In addition another new algorithm has been developed [6] which uses top down graph based approach. In addition, many research have been developed algorithms using tree structure, such as CT-ITL[5], FP-growth [7], AFP-Tree[9].

In this paper, we propose SMine algorithm for mining the set of all frequent itemsets in the database by reducing the number of scans. During the first database scan the number of occurrences of each item is determined and the infrequent ones are discarded. Then the frequent items are counted in each transaction. The transactions are sorted based on the number of frequent items in descending order. Then graph based approach is used to find the frequent item sets.

The organization of this paper is as follows: In Section 2, we put an insight into the detailed problem description. In Section 3, we give a detail of proposed SMine algorithm used for generating all frequent itemsets. Example is given in Section 4. We end with our conclusion in Section 5.

## 2. PROBLEM DESCRIPTION

Let I = $\{I_1, I_2, \ldots I_n\}$ be a set of items. Let D, the task-relevant data, be a set of transactions in a supermarket, where each transaction T is a set of items, such that $T \subseteq I$. Each transaction is assigned an identifier called TID. Let A be a set of items, a transaction T is said to contain A if and only if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$, and $A \cap B = \varnothing$. The rule $A \Rightarrow B$ holds in the transaction set D with support s, where s is the percentage of transactions in D that contain $A \cup B$ (i.e., both A and B). This is taken to be the probability $P(A \cup B)$. The rule $A \Rightarrow B$ has confidence c in the transaction set D if c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability, $P(B|A)$. That is, *Support* $(A \Rightarrow B) = P(A \cup B) = s$, *Confidence* $(A \Rightarrow B) = P(B|A)$ =Support $(A \Rightarrow B)$/Support (A)=c. Thus association rules is composed of the following two steps: 1) Find the large item sets that have transaction support above a minimum support and 2) From the discovered large item sets generate the desired association rules.

In this paper, we have developed a method to discover large item sets from the transaction database, thus finding a solution for the first sub problem.

## 3. PROPOSED METHOD

**Step 1:** The algorithm scans the database in order to count the number of occurrences of each item to find the candidate 1-itemset with their support count.

**Step 2**: The set of frequent 1-itemset $L_1$ can then be determined by removing the items having less than the minimum support count. It consists of the candidate 1-itemsets satisfying minimum support. Let the number of frequent 1-itemset be 'n'.

**Step 3:** Removes the infrequent items from each transaction and counts the number of items in each transaction ( item_count ).
**Step 4:** The transactions are sorted in descending order based on the item count.
**Step 5:** Create a table called 'M' with two columns such as 'no.of.items' and 'no.of. transactions'**.** Let no=n;
**Step 6:** Add a row with no.of.items = no and no.of.transactions is equal to the number of transactions having item_count >= no. If no >2, then decrement 'no' value by 1 and repeat step 6.
**Step 7:** Select the maximum 'no.of.items' from the table M having the 'no.of transactions' equal to or greater than the minimum support count. Let it be m. Create a directed graph structure starting from all possible items in m-itemset as the header nodes in the first level, all the possible (m-1) itemset in the second level, all the possible (m-2) itemset in the next level and So on, until 2-itemset.
**Step 8 :** Get the 'no.of transactions' for m-itemset from the table M. Let it be 'R'. If it is greater than or equal to the minimum support count, then find the support count of each unvisited node of the m-itemset by scanning first 'R' transactions. If the set is frequent, mark this node and all its sub nodes as frequent items. If the set is frequent, all its subsets must also be frequent.
**Step 9 :** Go to the next level .Let m=m-1. Repeat step 8 until m =2.
**Step 10:** All the marked nodes are frequent itemsets.

## 3.1 SMine Algorithm :

```
Input : Database, D, of transactions; minimum support
        threshold, min_sup.
Output : L, frequent itemsets in D.
Method:
Begin
    $L_1$= find_frequent_1-itemsets(D);
    n=number of items in $L_1$;
    $L_2$, $L_3$....., $L_n$ are initially set to null;
    for each transaction t in D
    {
        //Removing infrequent items from
        transactions
         Delete item not in $L_1$ from t .
          t. item_count = Number of frequent items.
    }
    Sort the transactions in D, descending order based on
    their item_count.
    call DB_details(D)
    call creategraph($L_1$, n)
End
```

```
procedure DB_details(D)
for( k=n; k>=2; k--)
      $M_k$=Number of transactions having item_count >=k;
// Arrange these values in a table called 'M'
 end procedure
```

```
procedure creategraph($L_1$, n )
// $L_1$ is frequent 1-itemset, n is the number of
// items in $L_1$
 while( $M_n$ satisfies min.support count)
          n=n-1;
Add all possible n- itemsets as the header nodes
$C_n$= all possible 'n' itemsets.
for ( k=n; k >= 3; k--)
      Call subgraph($C_k$)
for ( k=n; k >= 2; k--)
     Call frequent_items (k)
end procedure
```

```
procedure subgraph($C_k$)
 $C_{k-1}$ = {}
for each item in $C_k$
 {
        $C_{k-1}$ = $C_{k-1}$ U ( all the possible subset (k-1) itemsets )
        for each subset (k-1) itemset
    {
           If the node is already created then
                  make a link to the parent node
            else
                 create a new node for the item and make a link to
                  the parent node
     }
   }
end procedure
```

```
procedure frequent_items(k)
  for each itemset in $C_k$
 {
    If (item is not visited || not marked) then
     {
      Find the support count by scanning first $M_k$ transactions .
          If it satisfies the min. support then
        {
            Mark this node and all its subnodes up to the level
              2 itemsets are frequent.
            Add these marked nodes to $L_k$, $L_{k-1}$, $L_{k-2}$, …., $L_2$ in
              their respective array.
        }
        else
            mark this node as infrequent.
    }
 }
end procedure
```

## 4. EXAMPLE

Table 1 shows the transactional database with Transactional Identity Number(TID), List of item Identity numbers and number of items in each transaction.

**Table 1 : Transactional Database , D**

| TID | List of item_IDs | Item Count |
|---|---|---|
| T100 | I1, I2, I5,I6 | 4 |
| T200 | I2, I4 | 2 |
| T300 | I2, I3,I7 | 3 |
| T400 | I1, I2, I4 | 3 |
| T500 | I1, I3 | 2 |
| T600 | I2, I3 | 2 |
| T700 | I1, I3 | 2 |
| T800 | I1, I2, I3, I5 | 4 |
| T900 | I1, I2, I3 | 3 |

Scan the transactional Database, D for count of each Candidate items. It is shown in table 2.

**Table 2 : Candidate items, $C_1$**

| Item Set | Sup. Count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |
| {I6} | 1 |
| {I7} | 1 |

Compare the candidate support count with minimum support count and removes the infrequent items from Table 2 and the result is shown in Table 3

**Table 3 : Frequent 1-itemsets, $L_1$**

| Item Set | Sup. Count |
|---|---|
| {I1} | 6 |
| {I2} | 7 |
| {I3} | 6 |
| {I4} | 2 |
| {I5} | 2 |

Removing infrequent items from each transactions and update item_count and sort the transactions and it is shown in Table 4.

**Table 4 : Sorted Database**

| TID | List of item_IDs | Item_count |
|---|---|---|
| T800 | I1, I2, I3, I5 | 4 |
| T100 | I1, I2, I5 | 3 |
| T400 | I1, I2, I4 | 3 |
| T900 | I1, I2, I3 | 3 |
| T200 | I2, I4 | 2 |
| T300 | I2, I3 | 2 |
| T500 | I1, I3 | 2 |
| T600 | I2, I3 | 2 |
| T700 | I1, I3 | 2 |

Create a table called 'M' with two columns such as 'no.of.items' and 'no.of. transactions'**.**Add the rows with no.of.items and count the number of transactions having item_count >= no.of.items. It is shown in Table 5.

**Table 5: M**

| No. of items | No. of transactions | |
|---|---|---|
| 5 | 0 | M5 |
| 4 | 1 | M4 |
| 3 | 4 | M3 |
| 2 | 9 | M2 |

Select the maximum 'no.of.items' from table M having the 'no.of transactions' equal to or greater than the minimum support count. In this example m=3. Create a directed graph structure as shown in Figure 1, starting from all possible 3-itemsets as the header nodes in the first level, all the possible 2-itemset in the second level.

If the set is frequent, then all its subsets are also frequent. From the table 5, we find that $L_5$ and $L_4$ are null, because the number of transactions are less than the minimum support threshold. So, we can check the possible 3- itemsets.

It does not need to check the complete database. To find the frequent 3-itemset in the above example, it is enough to check first four transactions. If any 3-itemset is frequent, all its subsets are also frequent. Hence to find frequent 2-itemset, it is enough to check 5 patterns. The Frequent itemsets are as follows.

$L_1$={I1,I2,I3,I4,I5}
$L_2$={ {I1,I2}, {I1,I3}, {I2,I3}, {I2,I4}, {I1,I5}, {I2,I5}}
$L_3$ ={ {I1,I2,I3}, {I1,I2,I5}}

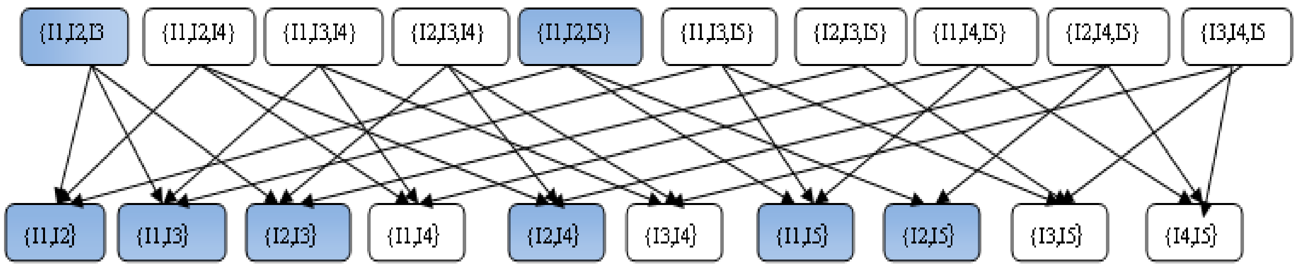

**Fig 1: Graph based approach**

## 5. CONCLUSIONS

SMine algorithm reduces the repeated scan of the complete database like Apriori. In this new algorithm only limited number of transactions are scanned starting from the first to find frequent n-itemset. It also uses the concept if the set is frequent, all its subsets are frequent. It is implemented through the graph based approach. We have explained this new algorithm and illustrated with examples. In our future work, we will implement and compare it with the existing pattern mining algorithms.

## 6. REFERENCES

[1] Ashok Savasere, E. Omiecinski and S. Navathe, "An efficient algorithm for mining association rules in large databases", Proceedings of the 21st International Conference on Very large database, 1995, pp. 420-431.

[2] Jia Ling, Koh and Vi-Lang Tu, " A Tree-based Approach for Efficiently Mining Approximate Frequent Itemsets", IEEE International Conference on Research Challenges in Information Science, 2010, pp. 25-36.

[3] Jian Pei ,J. Han, J. Lu, H. Nishio.S.and Tang, "H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases", ICDM International Conference on Data Mining, ICDM, 2001, pp. 441-448.

[4] Jiawei Han, Jian Pei, and Yiwen Yin, "Mining Frequent Patterns without Candidate Generation", Proceedings of ACM SIGMOD Conference, Dallas, TX, 2000, pp.53-87.

[5] Jong Soo Park, M.S. Chen, and P.S. Yu, "An effective hash based algorithm for mining association rules", Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25, 1995, pp. 175-188.

[6] Ramesh Agrawal and Ramakrishnan Srikant, "Fast algorithms for mining association rules", proceedings of the 20th VLDB Conference Santiago,Chille, 1994, pp. 487-499.

[7] Ramesh Agrawal, Tomasz Imielinski, and A. Swami, "Mining association rules between sets of items in large databases", ACM-SIGMOD Int. Conf. Management of Data, Washington, D.C., May 1993, pp 207–216.

[8] Senthil Kumar A.V and R.S.D. Wahidabanu, "A Frequent Item Graph Approach for Discovering Frequent Itemsets", Proceedings of 2008 IEEE International Conference on Advanced Computer Theory, 2008, pp.952-956.

[9] Yudho Giri Sucahyo and Gopalan.R, "Efficient Frequent Item Set Mining using a Compressed Prefix Tree with Pattern Growth", Proceedings of 14th Australian Database Conference, Adelaide, Australia, 2003, pp.95-104