A Search Space Reduction Algorithm for Mining Maximal Frequent Itemset

K.Sumathi
Assistant Professor,
Department of Computer Applications, K.L.N.C.I.T,
Madurai

S.Kannan, Ph.D
Associate Professor,
Department of Computer Applications,
MKU, Madurai.

K.Nagarajan
Chief Architect of Business Intelligence,
Tata Consultancy Services,
Chennai.

ABSTRACT
Abstract -Mining of frequent itemset plays important role in data mining applications. The algorithms which are used to generate the frequent patterns must perform efficiently. Because the overall performance of association rule mining based on fast discovery of frequent pattern. Many MFI approaches need to recursively construct many candidates, they also suffer the problem of a large search space, so that the performances for the approaches degrade when the database is massive or the threshold for mining frequent patterns is low. In this paper, an efficient method for discovering the maximal frequent itemsets is proposed which combines a vertical tidset representation of the database with effective pruning mechanisms for search space reduction. It works efficiently when the number of itemsets and tidsets are more. The proposed approach has been compared with GenMax algorithm for mushroom dataset and the results show that the proposed algorithm generates less number of candidate itemsets from which MFIs are obtained. Hence, the proposed algorithm performs effectively and generates maximal frequent patterns faster.

Keywords
Search space Reduction, Maximal Frequent Itemsets.

1. INTRODUCTION

Data mining or knowledge discovery in databases is a collection of exploration techniques based on advanced analytical methods and tools for handling a large amount of information. In data mining, association rule mininig is a popular and well researched method for discovering interesting relations between items in large transactional datasets. Association rule generation is usually split up into two separate steps. In the first step, minimum support is given by the user to find all frequent itemsets in a dataset. In the second step, these frequent itemsets and the minimum confidence constraint are used to generate rules. The support of an itemset X(sup(X)) is defined as the proportion of transactions in the data set which contain the itemset. sup(X)= no. of transactions which contain the itemset X / total no. of transactions. The confidence of a rule is defined as follows: conf(x->y)= sup(x∪y)/sup(x). Based on the different measures of interestingness, strong rules are discovered in datasets. Based on the perception of strong rules, association rules are introduced by Agrawal [7] for discovering regularities between products in large scale transaction data recorded in retail markets.
The problem of mining frequent itemsets has been a topic of Intensive research. Efficient algorithms for mining frequent items are crucial for mining association rules. Mining Frequent itemsets plays an important role in the field of data mining. Frequent itemsets are essential for many data mining problems, such as the discovery of association rules, data correlations and sequential patterns. The frequent pattern has several alternative forms including a simple frequent pattern, a closed pattern and maximal pattern. Frequent pattern is a pattern that satisfies a minimum support threshold. A pattern X is a closed pattern if there is no super pattern with the same support of X. A pattern X is a maximal pattern if there exists no frequent super pattern of X. once frequent patterns are mined, it can be mapped into association rules or other kind of rule based interestingness measures. Most existing work focuses on mining all frequent item sets (FI). However, since any subset of a frequent item set also is frequent, it is sufficient to mine only the set of maximal frequent item sets (MFI).

The frequent itemset mining problem can be formally stated as follows: Let I be a set of different items. Each transaction T in database D is a subset of I. if X is a set of items then $X \subseteq I$. An itemset with k items is called a k-itemset. The support of X, denoted by sup(X), is the number of transactions containing X. If sup(X) is greater than a user-specified minimum support, X is called a frequent itemset. An itemset X is closed frequent itemset if it has no proper superset with the same support. An itemset X is called a maximal frequent itemset if it has no proper superset that is frequent. An itemset X is a maximum length frequent itemset if X contains a maximum number of items in frequent itemset. Formally, it can be defined as follows: Let D be a transaction database over a set of different items I. Given a user-specified minimum support ε, an itemset X is a maximum length frequent itemset if $sup(X) \geq \varepsilon$ and for all itemset Y, if $sup(Y) \geq \varepsilon$ then the number of items contained in Y is greater than or equal to number of items contained in X.

The drawback of mining all frequent itemsets is that if there is a large frequent itemset with size n then almost all 2^n candidate subsets of the items might be generated. However, since frequent itemsets are upward closed, it is sufficient to discover only all maximal frequent itemsets (MFI's). The database representation is also an important factor in the efficiency of generating maximal patterns. In horizontal data format, the data is represented as tid-itemset format, where tid is the transaction identifier and itemset is the set of items included in the transaction. In vertical data format, the data is represented as item-tidset format, where item is the name of the item and tidset is the set of transaction identifiers containing the item. The vertical representation allows simple and efficient support counting.

2. RELATED WORKS

When the frequent patterns are long, mining all Frequent Itemsets is infeasible because of the exponential number of frequent itemsets. Thus algorithms for mining Frequent Closed Itemsets (FCI) are proposed, because FCI is enough to generate association rules. However FCI could also be exponentially large as the FI. However, since any subset of a frequent set also is frequent, it is sufficient to mine only the set of maximal frequent itemsets. Given the set of MFI, it is easy to analyze many interesting properties of the dataset, such as the longest pattern, the overlap of the MFI, etc. MFI mining has two advantages over all FI mining. First, MFI mines small and useful rules, and second a single database scan can collect all FI, if we have MFI. In this paper we present a new algorithm to find MFIs quickly by obtaining the possible Maximal frequent items (PMFIs) from which candidates are generated. So the number of candidates passed to the algorithm is relatively less when compared to existing algorithm. Some of the existing MFI mining algorithm is described below.

GenMax[2], is a efficient backtrack search based algorithm for mining maximal frequent itemsets. Genmax used number of optimizations to prune the search space. It used differential set propagation to perform fast frequency computation when the frequent items have more number of candidates and progressive focusing to perform maximal checking. Genmax reduce the number of superset checking by using progressive focusing technique of LMFI-Backtrack. GenMax is efficient method to mine the exact set of maximal patterns. Genmax uses vertical tidset format and computes the set of frequent items by counting the number of tids of each item and and frequent 2 items by intersecting the item's tids. Genmax uses itemset reorderting technique which reduces the search space of MFI tree.

Mafia is one of the recent methods proposed by Burdick, D., M. Calimlim and J. Gehrke,[1] for mining the MFI. The search strategy of the algorithm integrates a depth-first traversal of the itemset lattice with effective pruning mechanisms that significantly improve mining performance. Mafia implementation for support counting combines a vertical bitmap representation of the data with an efficient bitmap compression scheme. Mafia uses vertical bit-vector data format, and compression and projection of bitmaps to improve performance. Mafia uses three pruning strategies to remove non-maximal sets. The first is the look-ahead pruning which is introduced in MaxMiner. The second is to check if a new set is subsumed by an existing maximal set. The last technique checks if $t(X) \subseteq t(Y)$. If so X is considered together with Y for extension. Mafia mines a superset of the **MFI**, and requires a postpruning step to eliminate non-maximal patterns.

Max Miner [6] is another algorithm for finding the maximal elements. It uses efficient pruning techniques to quickly narrow the search. Max Miner employs a breadth first traversal of the search space. It reduces database scanning by employing a look ahead pruning strategy. For frequency computation maxminer use a additional technique called support lower bounding. This algorithm also uses dynamic reordering technique to reduce the size of the search space. This process produces small number of frequent extension for the next level. The item which occurs in the fewest number of large itemsets should be occurred first and item occurring in the maximum number of large itemset should be occurred last.

DepthProject[5] finds long itemsets using a depth first search of a lexicographic tree of itemsets, and uses a counting method called bucketing based on transaction projections along its branches. DepthProject uses a horizontal database layout and use some form of compression when the bitmaps become sparse. DepthProject also uses the look-ahead pruning method with item reordering. It returns a superset of the **MFI** and would require post-pruning to eliminate non-maximal patterns.

The Pincer-Search [3] algorithm uses horizontal data format and hybrid approach to mine Maximal frequent patterns. It constructs the candidates in a bottom-up and top-down direction at the same time, maintaining Maximal Frequent Candidate Patterns. It generates candidates and infrequent items in bottom up direction and based on the infrequent item MFCS are splitted into more than one candidate. This can help in reducing the number of database scans, by eliminating non-maximal sets early. The MFCS are supersets of the maximal frequent itemsets. Hash-Based Method HMFS [4] generates the maximal frequent itemsets in the category of the combination of bottom-up and top-down search. This method combines the advantages of both the DHP and the Pincer-Search algorithms. The HMFS method reduces the number of database scans when the length of the longest frequent itemset is relatively long. The HMFS method obtains the infrequent itemsets with the hash technique from the bottom-up direction and then can use the filtered itemsets to find the maximal frequent itemsets in the top-down direction. HMFS uses the hash technique of the DHP algorithm to filter the infrequent itemsets in the bottom-up direction and uses a top-down technique that is similar to the Pincer-Search algorithm to find the maximal frequent itemsets.

3. PROPOSED WORK

The proposed approach focuses on Mining Maximal Frequent Itemset. In this paper, a backtracking method and effective pruning mechanism to reduce the search space is proposed for generating Maximal frequent patterns. Backtracking method is first introduced by Karam Gouda and Mohammad Zaki in GenMax algorithm to generate Maximal Frequent Itemsets. Backtracking method is used here to generate Maximal Frequent Itemsets from PMFIs.

There are two main factors to develop an efficient MFI algorithm. The first is the database representation used to perform fast frequency computations, and the second is the set of techniques used to reduce the size of search space. Here, we are using vertical data format for storing the transactions in the database to compute the frequency of an itemset quickly. The vertical representation has the following major advantages over the horizontal layout: Firstly, computing the support of itemsets is simpler and faster with the vertical layout since it involves only the intersections of tidsets. Secondly, with the vertical layout, there is an automatic "reduction" of the database before each scanning that only those itemsets that are relevant to the following scan of the mining process are accessed from disk.

Most of the Maximal frequent itemset mining algorithm sends all frequent items as candidate items to the recursive algorithm to obtain MFIs. The size of the search space depends on number of candidate items to be sent to the algorithm. Because N candidate generates 2^N nodes at worst case while constructing the MFI Tree. In this paper Search Space Reduction (SSR) algorithm is proposed to mine MFIs. To prune the search space of MFI tree the number of

candidates passed to the algorithm must be reduced. In this approach PMFIs are generated once frequent item and candidates of each frequent item are obtained. All candidates are generated from these PMFIs. So number of candidate passed to the algorithm is relatively small which in turns reduce the search space of MFI tree.

Consider our example database which includes five different items, I = {A, C, D, T, W} and six transactions T= {1, 2, 3, 4, 5, 6}. The vertical data format of the database d is given below.

Item	Tidset
A	T1, T3, T4, T5
C	T1,T2,T3,T4,T5,T6
D	T2,T4,T5,T6
T	T1,T3,T5,T6
W	T1,T2,T3,T4,T5

Table 1 : Vertical Data format of the transactional database d.

All Frequent items are extracted and reordered in ascending order with respect to the support .The support is directly given by the number of transactions in the tidset of each item. Let us consider the minimum support to be 3. From the above structure, all items are frequent. The items A, C, D, T and W are reordered in ascending order with respect to the support and these are considered to next level. The Frequent 1 items are A, D, T, W and C.

In the next level PMFIs are constructed by obtaining candidate sets for each frequent item. All candidates are reordered in increasing order of support.

All PMFIs are sorted with respect to their size in descending order. In database d, the PMFIs are ATWC, DWC, TWC, WC and C. Each PMFI is checked whether it has no superset in MFI or not. If PMFI has no superset in MFI then it is checked whether it is frequent. If it is, then the PMFI is added to MFI directly.

A PMFI can be added to MFI directly if it has no superset in MFI and is frequent. This early finding of MFI relatively reduces the number of candidates passed to the algorithm.

In this example the first PMFI is ATWC which has no superset in MFI and it is frequent. So it is added to MFI directly. The next PMFI is DWC which has no superset in MFI and it is frequent and added to MFI directly. The subsequent itemsets TWC, WC, and C are having superset in MFI (ATWC) and are ignored. If DWC is infrequent then it is sent to the GenerateMFI algorithm to mine MFIs from DWC.

Frequent Item	**Candidate sets**	**PMFIs**
A	**C ,T, W**	**ATWC**
D	**W,C**	**DWC**
T	**W,C**	**TWC**
W	**C**	**WC**
C	**-**	**C**

Table 2 : PMFIs of frequent item in database D.

This algorithm also obtain the size of largest PMFI and cutoff which is sqrt(number of frequent itemsets). If the size of largest PMFI is less than the cutoff then the infrequent PMFIs are passed to the GenerateMFI algorithm directly to mine MFI from PMFI. Because the size of PMFIs will be very small and instead of form a candidate set, each PMFI is passed to the algorithm. If the size of largest PMFI is greater than the cutoff then the infrequent large PMFIs are merged into candidate sets and it is passed to the algorithm. GenerateMFI algorithm returns a superset of the **MFI** and would require post-pruning to eliminate non-maximal patterns.

SSR Algorithm

1. Generate Frequent 1 items and reorder them in ascending order of their support.
2. Construct PMFIs by obtaining candidate items (reorderd in increasing order of support) for each frequent item.
3. Sort PMFIs with respect to their size in descending order.
4. MaxsizePMFI=size of Largest PMFI , cutoff=sqrt(number of frequent itemsets)
5. For each x ε PMFIs
6. If x has no superset in MFI
    a. If size of x is 1 or 2 then add x to MFI
       // most of the sparse dataset the candidate items of frequent item may be 1 or 2;
    b. If x is frequent then add x to MFI;
    c. Else if (MaxsizePMFI > cutoff)
    d. generateMFI(x.freq,x.cand,(x.freq).tid)
    e. else
    f. candidate=candidate ∪ x
7. reordercandidate;
8. generateMFI(empty,candidate,empty)

generateMFI Algorithm

```
generateMFI (frequent ,candidate, ftids)
{
For each x ε candidate
        If frequent ∪ candidate has superset in
MFI then returns
Nfrequent={}
Nfrequent.add(frequent U x)
        If ftids=={}
        Ntids=x.tid;
        Else
```

```
        Ntids=x.tid ∩ ftids
candidate.remove(x)
        if candidate is empty && Nfrequent has
no superset in MFI
        add Nfrequent to MFI
newcandidate =
generatecandidates(Nfrequent,candidate,Ntids)
        If newcandidate is empty
        if Nfrequent has no superset in MFI
                add Nfrequent to MFI
        Else
        generateMFI (Nfrequent,
newcandidate,Ntids)
}

generatecandidates(frequent,candidate,ftids)
{cand=null;
for each x ε candidate
        If(ftids ∩ tid(x) ≥ support)
        cand.add(x); // candidates are stored in
increasing order of support.
return(cand);
}
```

Pruning

Most of the standard algorithm like mafia, depthproject, genmax takes all frequent items as candidates and MFIs are found from these candidates. To reduce the search space of the MFI tree the number of candidates passed to the algorithm is to be reduced. We propose a pruning technique to achieve the same. Once FIs are generated, they are reordered with respect to its support in ascending order and this technique is introduced by Roberto Bayardo in MaxMiner algorithm for mining maximal frequent itemsets. Once frequent items are generated, the candidates of each frequent item are obtained. The Frequent items and Candidates of the frequent items are combined to form PMFIs and they are sorted in descending order of their size. PMFIs can be directly added to MFI if they are frequent and no superset in MFI. Every PMFI is checked whether it is MFI or not, if it is MFI then it is not passed to the algorithm. So we can find a MFI as early as possible.

Definition1: PMFIs- The combination of frequent item and it's reordered candidates is called PMFI (Possible maximal frequent itemset). For example if A is the frequent item and candidates of A are T, W, C then the PMFI is ATWC.

Definition2: if the PMFI is frequent then it is added to MFI directly which in turns reduce the search space of the MFI tree. ATWC (PMFI) is frequent then it can be added to the MFI directly.

Definition3: PMFIs are not included in the candidate set if it becomes MFI. The PMFI, ATWC is frequent so it is not added to the candidate set. If x is an infrequent PMFI then it is added to the candidate set. All infrequent PMFIs are combined to form a candidate set and it is reordered in ascending order with respect to the support and passed to the GenerateMFI algorithm to mine the remaining MFIs.

4. RESUTS

The testing of the proposed algorithm has been carried out on the real dataset (containing long itemsets) Mushroom. The number of candidates passed to the algorithm to mine MFIs by the SSR algorithm is compared to Genmax algorithm for various values of minimum support. The SSR algorithm has been compared with GenMax algorithm and results show that the SSR algorithm passes less number of candidate itemsets from which MFIs are mined.

Figure 1 illustrates that, the SSR algorithm passes less number of candidate itemsets and may have better performance when compared to conventional GenMax algorithm. Support is taken as x axis and the number of candidate itemsets taken to find MFI is taken as y axis.

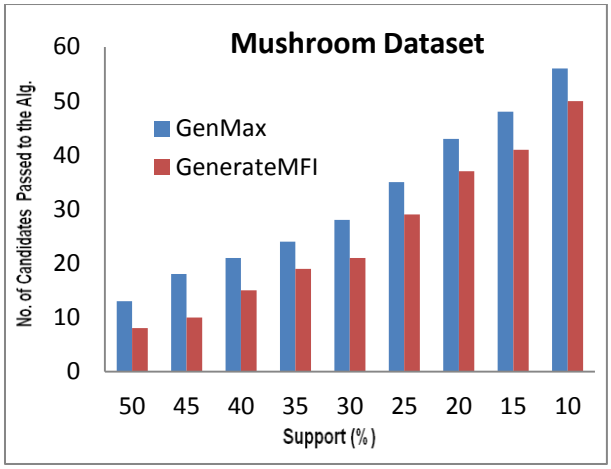

Figure 1. Number of Candidate itemsets passed to GenerateMFI and GenMax Algorithm from Mushroom dataset.

5. CONCLUSION

In this paper we have introduced a pruning technique for search space reduction. The initial candidates for every frequent item are generated by finding association among frequent items. PMFIs are generated and sorted in descending order of their size. PMFIs are added to MFI directly if the PMFI is frequent. So that the number of candidates passed to the algorithm is relatively reduced. The infrequent PMFIs are combined to form a candidate set and it is passed to the generateMFI algorithm to mine remaining MFIs.

6. REFERENCES

[1] Burdick, D., M. Calimlim and J. Gehrke, "MAFIA: A maximal frequent itemset algorithm for transactional databases", In International Conference on Data Engineering, pp: 443 – 452, April 2001, doi = 10.1.1.100.6805

[2] K. Gouda and M.J.Zaki, "Efficiently Mining Maximal Frequent Itemsets", in Proc. of the IEEE

[3] D. Lin and Z. M. Kedem, "Pincer-Search: A New Algorithm for Discovering the Maximum Frequent Set", In Proceedings of VI Intl. Conference on Extending Database Technology, 1998.

[4] Don-Lin Yang, Ching-Ting Pan and Yeh-Ching Chung An Efficient Hash-Based Method for Discovering the Maximal Frequent Set

[5] Agrawal, R., Aggarwal, C., and Prasad, V. 2000. Depth first generation of long patterns. In 7th Int'l Conference on Knowledge Discovery and Data Mining, pp. 108–118.

[6] Roberto Bayardo, "Efficiently mining long patterns from databases", in ACM SIGMOD Conference 1998.

[7] R. Agrawal, T. Imielienski and A. Swami, "Mining association rules between sets of items in largedatabases. In P. Bunemann and S. Jajodia, editors, Proceedings of the 1993 ACM SIGMOD Conference on Management of Data, Pages 207-216, Newyork, 1993, ACM Press.

[8] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, "Fast discovery of association rules", Advances in Knowledge Discovery and Data Mining, pages 307-328, MIT Press, 1996.

[9] Tianming Hu,Sam Yuan Sung b, Hui Xiongc, Qian Fud , Discovery of maximum length frequent itemsets, *Journal of Information Sciences 4 February 2007*, http://datamining.rutgers.edu/publication/ins2008.pdf

[10] Jiawei Han, Hong Cheng, Dong Xin , Xifeng Yan, Frequent pattern mining: current status and future Directions, http://www.cs.ucsb.edu/~xyan/papers/dmkd07_frequentpattern.pdf