Discovering Maximal Frequent Item set using Association Array and Depth First Search Procedure with Effective Pruning Mechanisms

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

S.Kannan, PhD.
Associate Professor,
Department of Computer Science,, MKU,
Madurai.

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

ABSTRACT

The first step of association rule mining is finding out all frequent itemsets. Generation of reliable association rules are based on all frequent itemsets found in the first step. Obtaining all frequent itemsets in a large database leads the overall performance in the association rule mining. In this paper, an efficient method for discovering the maximal frequent itemsets is proposed. This method employs Association array technique and depth first search technique to mine Maximal Frequent Itemset. The proposed algorithm GenMFI takes vertical tidset representation of the database and removes all the non-maximal frequent item-sets to get exact set of MFI directly. Pruning is done for both search space reduction and minimizing the number of frequency computations and number of maximal frequent candidate sets.

The algorithm gives better results for the sparse dataset even though number of the Maximal Frequent Itemset is huge. The proposed approach has been compared with Pincer search algorithm for T10I4D100K dataset and the results shows that the proposed algorithm performs better and generates maximal frequent patterns faster. In order to understand the algorithm easily, an example is provided in detail.

Keywords

Mining Maximal Frequent Itemsets –Association Array, Depth First Search, Pincer search algorithm.

1. INTRODUCTION

The association rule mining task is a two step process. In the first step all frequent itemsets are obtained. This is both computation and I/O intensive. Given m items there can be potentially 2^m frequent item sets. It constitutes an area where significant research findings have been reported. In the second step confident rules are generated – Rules of the form X/Y ⇒ Y where Y ⊂ X are generated for all frequent itemsets obtained in step I provided they satisfy the minimum confidence. Our focus is on the generation of frequent itemsets. the problem of association rule mining is defined as: Let I = { i1,i2,i3,…..in} be a set of ℕ binary attributes called items. Let D={t1,t2,t3…tm}be a set of transactions called the database. Each transaction in D has a unique transaction ID and contains a subset of the items in I. A rule is defined as an implication of the form x =>ywhere X,Y ⊆ I and X ∩Y= ¢. The sets of items $X$and Y are called antecedent and consequent of the rule respectively. In a transactional database, set of items is I={milk, bread, butter, beer}. An example rule for the supermarket could be {butter, bread}=> {milk} meaning that if butter and bread are bought, customers also buy milk. A frequent itemset is one that occurs in at least a user-specific percentage of the database. That percentage is called support. An itemset is closed if none of its immediate supersets has the same support as the itemset. An itemset is maximal frequent if none of its immediate supersets is frequent. Downward Closure Property (Basis for Top-down Search): states that "If an itemset is frequent then all its subsets must be frequent."Upward Closure Property (Basis for Bottom-up Search): states that "If an itemset is infrequent then all its supersets must be infrequent."

A major challenge in mining frequent patterns from a large data set is the fact that such mining often generates a huge number of patterns satisfying the min_sup threshold, especially when min_sup is set low. This is because if a pattern is frequent, each of its subpatterns is frequent as well. when a transaction database contains large number of large frequent itemsets, mining all frequent itemsets might not be a good idea. As an example, if there is a frequent itemset with size l, then all 2^l nonempty subsets of the itemset have to be generated. Thus, a lot of work is focused on discovering only all the maximal frequent itemsets (MFI's). Unfortunately, mining only MFI's has the following deficiency. From an MFI and its support s, we know that all its subsets are frequent and the support of any of its subset is not less than s, but we do not know the exact value of the support. To solve this problem, another type of a frequent itemset, the Closed Frequent Itemset (CFI), has been proposed. In most cases, though, the number of CFI's is greater than the number of MFI's, but still far less than the number of FI's.

A large pattern will contain an exponential number of smaller, frequent sub-patterns. To overcome this problem, closed frequent pattern mining and maximal frequent pattern mining were proposed. A pattern α is a closed frequent pattern in a data set D if α is frequent in D and there exists no proper super-pattern β such that β has the same support as α in D. A pattern α is a maximal frequent pattern (or max-pattern) in setD if α is frequent, and there exists no super-pattern β such that α ⊂ β and β is frequent in D.

The First step of frequent mining process is to compute the support count of each and every item in the database to extract frequent items. This process requires accessing the database. The database representation is also an important factor in the efficiency of generating and counting itemsets. In general

Items in the database can be stored in memory in the following ways. Dataset organizations are typically horizontal or vertical. The main difference between these two approaches is, in case of horizontal representation the computation of support count of an itemset requires complete scan of database, whereas in vertical representation the support count of an itemset is calculated by intersection of transaction sets of items in itemset

2. RELATED WORKS

The Apriori algorithm [7] is a standard algorithm for finding frequent itemsets and most of algorithms are its variants. It uses frequent itemsets at level k to explore those at level k + 1, which needs one scan of the database. it employs the heuristic that all nonempty subsets of a frequent itemset must also be frequent, which prunes unpromising candidates to narrow down the search space. Apriori is based on the horizontal format of the database representation, in which a transaction is represented as an item list. An alternative way is to represent a database in vertical format, i.e., each item is associated with a set of transaction identifiers (TIDs) that include the item. As a representative in this group, VIPER [8] uses a vertical bit-vector with compression to store intermediate data during execution and performs counting with a vertical TID-list approach.

The Pincer-Search [1] algorithm uses horizontal data format. It constructs the candidates in a bottom-up and top-down direction at the same time, maintaining a candidate set of maximal patterns. This can help in reducing the number of database scans, by eliminating non-maximal sets early. The maximal candidate set is a superset of the maximal frequent itemset. The Pincer-Search algorithm scans a database to count the supports of all itemsets in MFCS and Ck in the bottom-up and top-down directions and classifies all itemsets in Ck into two groups, frequent and infrequent, in the bottom-up direction. The group that contains all frequent itemsets is Lk. The other group that contains all infrequent itemsets will be used to split the maximal frequent candidate itemsets in MFCS in the top-down direction. The algorithm will be terminated when there are no itemsets in MFCS. The Pincer-Search algorithm also uses the downward closure. The downward closure consists of two properties. The first property is that all supersets of the infrequent itemsets must also be infrequent. This property is used in many typical bottom-up algorithms of the association rule mining, such as the Apriori algorithm. The second property is that all subsets of a frequent itemsets must also be frequent. This property can be used in a top-down algorithm of the association rule mining. The Pincer-Search algorithm is very efficient when the length of the longest frequent itemset of a database is long. However, its disadvantage is that the initialization of the maximal frequent candidate set is not efficient. It may spend a lot of time on finding the set of maximal frequent itemsets.

Hash-Based Method HMFS[2] 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. Unlike the DHP algorithm, the HMFS method reduces the number of database scans when the length of the longest frequent itemset is relatively long. The HMFS method filters 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, combines the advantages of both the DHP and Pincer-Search algorithms. 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.

MaxMiner [4] is another algorithm for mining the maximal frequent itemsets. It uses efficient pruning techniques to quickly narrow the search. MaxMiner employs a breadth-first traversal of the search space; it uses a lookahead pruning strategy to reduce the database scan. It also employs item (re)ordering heuristic to increase the effectiveness of superset-frequency pruning. Since MaxMiner uses the original horizontal database format, it can perform the same number of passes over a database.

GenMax[3] is a effective algorithm by Gouda and Zaki employs a back track search technique for finding maximal itemset patterns . GenMax introduced a novel concept for finding supersets in the maximal frequent patterns called progressive focusing. GenMax also uses diffset propagation for fast support counting dynamic reordering of itemset to reduce the size of search space. Genmax uses the original vertical database format.

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.

Mafia[6] is also an important method for mining the **MFI**. 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 first used 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.

3. DISCOVERY OF MAXIMAL FREQUENT ITEMSET

The bottom-up approach gives better results when all maximal frequent itemsets are short and the top-down approach gives better results when all maximal frequent itemsets are long. In some datasets, some of the MFI are long and some are short. In this scenario both one-way search approaches will not be efficient.

The proposed method is very efficient in reducing the number of database scans and search space. This method can filter the infrequent itemsets in the bottom up direction and use the infrequent itemsets to find the maximal frequent itemsets in the top-down direction. In addition, this method also provides an efficient mechanism to construct the maximal frequent candidate itemsets.

The proposed approach focuses on Mining Maximal Frequent Itemset Generation. In this paper, Association Array is used to discover the frequent2 items and Initial MFCS and effective pruning Mechanism is used for generating Maximal frequent sets from IMFCS. There are two main ingredients to develop an efficient MFI algorithm. The first thing is techniques used to reduce the size of search space, and the second is the

representation used to perform fast frequency computations. This paper describes how proposed algorithm GenMFI achieves the same.

GenMFI algorithm generates all Maximal Frequent Itemset from dataset in four steps. In the first step all frequent items are generated. This approach takes vertical data format of the database so support need not be calculated separately. Fast frequency computation is done by intersecting transaction Ids(tids) of itemsets. In this case, support is directly given by the number of transactions in the tidlist of each FI. In second step association array is created by obtaining association between every two item. All infrequent 2 items and Initial Maximal Frequent itemset are discovered from this association array. In third step MFCS are generated by removing all infrequent-2 items from IMFCS. Itemsets are stored in MFCS in ascending order with respect to their size. In forth step all MFIs are obtained from MFCS using FindMFI algorithm. The algorithm is described with the following example.

Consider our example database which includes six different items, I = {A, B, C, D, E, F} 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, T2, T3, T4, T5
B	T2, T5, T6
C	T1, T2, T4, T5, T6
D	T1, T3, T4, T5
E	T2, T3, T4, T5, T6
F	T2, T4, T6

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

All Frequent items are extracted first. 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, B, C, D, E and F are frequent items and will be considered to next level.

In the next step Association Array(AA) is constructed by finding association between every two items. The Initial MFCS(IMFCS) and Infrequent 2 items are obtained from the association array. From the AA infrequent 2 sets are AB, AF, BD, BF, DF . IMFCS denotes the possible extension of each frequent items. for example frequent items A has possible candidates C,D,E. Initial MFCS are ACDE, BCE, CDEF, DE and EF.

AA	A	B	C	D	E
F	0	0	1	0	1
E	1	1	1	1	
D	1	0	1		
C	1	1			
B	0				

Figure 1 : Association Array for database D

In the third step, MFCS are generated from IMFCS by removing infrequent Items. if itemset in MFCS has no infrequent itemset then it is directly added into MFCS.A maximal candidate frequent itemset can't be a Maximal frequent itemset, if it includes any infrequent temsets. For example the third IMFCS is CDEF, has an infrequent itemset DF. So this IMFCS cannot be a MFI. When an itemset x in IMFCS has infrequent itemset y, replace every item x by |y| itemsets each obtained by removing y as a single element of x. it can be split into two itemset CDE and CEF. These itemset are not having any infrequent 2 itemsets and are added to the MFCS. Itemsets in MFCS are stored in ascending order with respect to the itemset size. The MFCS are DE, EF, BCE, CDE, CEF and ACDE. Algorithm is described below.

Algorithm

Algorithm : FindMFI

Input : MFCS and Support

Output : MFI

findMFI(x, sup)

if x has superset in MFI then return;

if x is frequent

if x has no superset in MFCS then addToMFI(x);

else

for each e in superset of x in MFCS

MFCS= MFCS/{e}

findMFI(e,sup);

If x has no superset in MFI

addToMFI(x);

else

if x has superset in MFCS

updateMFCS(x);

splitMFCS(x);

else

splitMFCS(x);

Algorithm: updateMFCS

Input : Old MFCS and infrequent set X

Output: New MFCS

1. x1=x
2. For all itemset m ε superset(x) in MFCS
3. MFCS=MFCS \ {m}
4. If x and m starts with same item
5. x1=x \ {firstitem of x}
6. for all item e ε x1
7. If m/{e} has superset in MFI or m/{e} has superset starts with first item of m in MFCS
8. Continue;
9. else
10. MFCS=MFCS U {m\{e}}
11. return MFCS

Algorithm: splitMFCS

Input : Infrequentset x

Output: New MFCS

1. for all item e ε x

2. If x/{e} has superset in MFI or {x/e} has superset starts with first item of x in MFCS

3. Continue;

4. else

5. MFCS=MFCS U {x\{e}}

6. return MFCS

In forth step all MFIs are generated from MFCS by using the findMFI algorithm. Each MFCS is passed to the algorithm to obtain MFIs and the algorithm will be terminated when there are no itemsets in MFCS.

1. DE is frequent and has superset in MFCS named CDE. Algorithm is invoked with CDE and it is an infrequent item set and it has one superset in MFCS (ACDE). CDE is split into CD, CE. CE has a superset in MFCS (CEF)(CEF and CE has the first item C) and it is ignored. ACDE is also split into ACD, ACE, and ADE. MFCS is updated by adding the new itemsets ACD, ACE, ADE, and CD. DE has no superset in MFI and has one superset (ADE) in MFCS. ADE is frequent and no superset in MFI and added to MFI and now DE has superset in MFI and no superset in MFCS and it is ignored.

2. The next itemset in MFCS is CD, it has superset in MFCS named ACD which is also frequent and no superset in MFI and is added to MFI and now CD has superset in MFI and ignored.

3. The next ietmset in MFCS is EF, it has superset in MFCS named CEF which is also frequent and no superset in MFI and is added to MFI and now EF has superset in MFI and ignored.

4. BCE is frequent and has no superset in MFCS and in MFI and it is added to MFI.

5. ACE is frequent and it has no superset in MFCS and in MFI and it is added to MFI.

Updating the MFCS

MFCS are sorted in ascending order with respect to their size. Infrequent itemset in MFCS require modification of MFCS. When an itemset x is identified as infrequent all supersets of x are removed from MFCS. Every superset y is replaced by |x| itemsets each obtained by removing y as a single element of x, when the infrequent itemset is not start with the first item of y and no superset in MFI.

For example CDE is identified as infrequent itemset and it has a superset (ACDE) in MFCS. ACDE is removed from MFCS and new MFCS are generated from this itemset by removing a single element of CDE. The itemsets are ADE, ACD, and ACE. If these itemsets have no superset in MFCS and MFI then added to MFCS.

If the itemset is infrequent and it has superset in MFCS and both starts with the same item then the first item of infrequent itemset is removed and this new infrequent itemset is used to split the superset. For example ACE is infrequent and it has one superset (ACDE) in MFCS. The infrequent itemset and superset of ACE start with the same item A. the first item of infrequent itemset A is removed and the new infrequent itemset is CE. CE is used to find new MFCS from ACDE. The itemsets are ACD, ADE. The infrequent itemset x is also split into (n-1) itemset (where n is the size of itemset) must include the first item of x (AC, AE). Split is done if the size of itemset is greater than or equal to three.

Reducing number of candidates

Database representation plays major role in frequency computation. Here we are using vertical representation (item, tidset format) of database. When vertical database format is used, the support of an item is calculated by counting the number of transaction in its tidset and itemset is calculated by counting the number of transaction after intersecting the transactions of item in itemsets. Frequency of an itemset x is computed if x has no superset in MFI.

Pruning is done in one direction based on the information gathered in the search in other direction. The search starts with smallest itemset in MFCS and continues until there is no itemset in MFCS. if an infrequent itemset X is found in MFCS it can be used to eliminate supersets of X by splitting supersets into n subsets (n is number of item in x). When maximal frequent itemset X is found in MFCS then this itemset can be used to eliminate subsets of X from MFCS. This approach can utilize both upward and downward properties and thus speed up the search for mining the maximum frequent set.

The bottom-up approach use upward closure property to reduce the number of candidates. The top-down approach use downward closure property to reduce the number of candidates. In this method Association Array is constructed to reduce the number of maximal candidate itemsets, because all MFCS are generated only from IMFCS by removing infrequent two itemsets. The proposed algorithm use both downward closure and upward closure properties to prune candidates and employ depth first search to effectively find maximal frequent itemsets from MFCS.

4. RESULTS AND DISCUSSIONS

The algorithm finds the Maximal frequent itemsets not only in the bottom-up direction but also in the top-down direction. The execution time is improved since the reduction of the number of candidates can significantly decrease both I/O time and CPU time.

The GenMFI algorithm has better performance than the Pincer-Search. There are two reasons. First, proposed method use Association Array to filter the huge infrequent 2-itemsets instead of actually counting the supports of all 2-itemsets Second, MFCS are generated from IMFCS and updated by genMFI algorithm instead of the combination of all distinct 1-itemsets in a database. The approach reduces the search space substantially. The disadvantage of pincer search algorithm is initialization of the maximal frequent candidate set is not efficient. It may spend a lot of time on finding the set of maximal frequent itemsets. Association array is used to generate MFCS which in turns reduce the search space.

The testing of the proposed algorithm has been carried out on the dataset T10I4D100K, it is measured that, the number of candidate itemsets and frequency computation taken by the proposed algorithm to find MFIs and it is compared to Pincer Search algorithm for various values of minimum support. The support percentage is varied from 2 to 0.8. The results show that the proposed algorithm generates MFIs very quickly than our implementation of Pincer Search Algorithm.

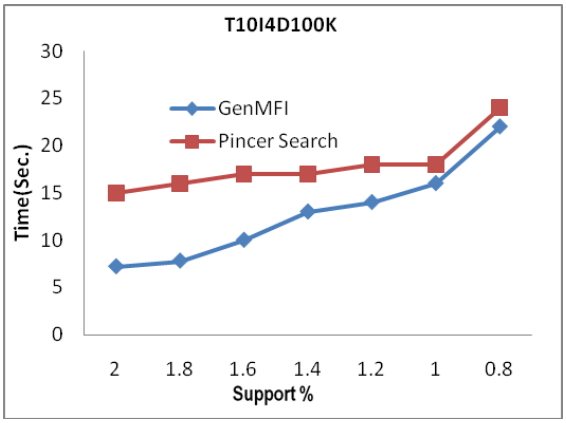

Figure 2. Execution time comparison of GenMFI algorithm and PincerSearch on T15I6D100K dataset.

5. CONCLUSION

In this paper, a new algorithm for discovering the maximal frequent itemsets is proposed. Association array is used to find the association between every two item and generate the IMFCS. Advantage of this method, is reduce the number of database scans by constructing MFCS from IMFCS. This method can filter the infrequent itemsets and can use the filtered itemsets to find the Maximal frequent itemsets. In addition, an efficient depth first search mechanism generates the maximal frequent itemsets from MFCS is provided. The experimental results show that our method has better performance than the Pincer-Search algorithms.

6. REFERENCES

[1] 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.

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

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

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

[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] 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

[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.