An Innovative Approach for finding Frequent Item sets using Maximal Apriori and Fusion Process and its Evaluation

Shailendra Chourasia
M.Tech, Scholar, Gyan Ganga College of Technology, Jabalpur (M.P.) Pin-482003, India

Rashmi Vishwakarma
Lecturer, MCA Department, Gyan Ganga Institute Of Technology& Sciences, Jabalpur, (M.P.)Pin-482003,India

Neeraj Shukla
HOD, CSE Department
Gyan Ganga College of Technology, Jabalpur, (M.P.)Pin-482003 , India

Meghna Utmal
HOD, MCA
Gyan Ganga College of Technology, Jabalpur, (M.P.)Pin-482003 , India

ABSTRACT
Frequent pattern mining is a vital branch of Data Mining that supports frequent itemsets, frequent sequence and frequent structure mining. Our approach is regarding frequent itemsets mining. Frequent item sets mining plays an important role in association rules mining. Many algorithms have been developed for finding frequent item sets in very large transaction databases. This paper proposes an efficient SortRecursiveMine (Sorted and Recursive Mine) Algorithm for finding frequent item sets. This proposed method reduces the number of scans in the database by first finding the maximal frequent itemsets in the database and then all its subset consider as frequent according to Apriori property. Then reduce the database by just considering only those transactions which are 1-Itemset frequent but not contain in frequent itemsets and then mine the remaining left frequent itemsets. Our proposed SortRecursiveMine algorithm works well based on recursive condition. Thus it reduces the memory constraints and helps to efficiently mine frequent itemsets in less time. At last we are evaluating this method, and performed an experiment on a real dataset to test the run time of our proposed algorithm.

Key words

Data Mining, Frequent Itemsets, Apriori Algorithm, FP-Growth, SortRecursiveMine Algorithm.

1. INTRODUCTION
Today's data mining research the association rule mining is a broad area. It consists of two phases i.e., finding of frequent itemsets and generation of rules from the revealed frequent itemsets. Finding frequent itemsets is more popular because it has many numbers 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] is 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 H-mine[3], FP-growth [7], AFP-Tree[9].

This paper proposes an efficient SortRecursiveMine (Sorted and Recursive Mine) Algorithm for finding frequent item sets. This proposed method reduces the number of scans in the database by first finding the maximal frequent itemsets in the database and then reduce the database by just considering only those transactions which are frequent 1-Itemset but not include in frequent itemsets and then mine the remaining left frequent itemsets. Our proposed SortRecursiveMine algorithm reduces the memory constraints and helps to efficiently mine frequent itemsets in less time.In Section 2, we put an insight into the detailed problem description. In Section 3, we give a detail of proposed SortRecursiveMine algorithm used for generating all frequent itemsets. Example is given in Section 4. We end with our conclusion in Section 7.

2. PROBLEM DESCRIPTION
Let I = {M_1, M_2, ….Mn} be a set of items. Let R, the task relevant data, be a set of transactions in a shop, where each transaction T is a set of items, such that T$\subseteq$ I. Each transaction is assigned an unique 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= ϕ. The rule A $\Rightarrow$ B holds in the transaction set R with support s, where s is the percentage of transactions in R 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 R if c is the percentage of transactions in R 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$\cup$ 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

Input : Transactional Database D, minimum support count.

Step 1: Count the number of occurrences of each item to find the candidate 1-itemset with their support count by scanning the database.

Step 2: Generate frequent 1-itemset by removing the items having less support count then minimum support count, from candidate 1-itemset. Let the number of frequent 1-itemset be "n".

Step 3: Removes the infrequent items from each transaction.

Step 4: Counts the number of items in each transaction (item_count).

Step 5: The transactions are sorted in descending order (i.e. the name of sorted database is SDatabase) based on the item_count.

Step 6: Call SortRecursiveMine(SDatabase).

Step 7: Stop.

3.1 SortRecursiveMine(SDatabase) Procedure

Step 1: Make an array with 2-dimension; then put transaction into and their respective count of repetition.

Step 2: Find maximal transactions (k-itemset) from the array whose count is greater than or equal to the minimum support known as maximal frequent itemsets or transactions. If k-itemsts count is less than minimum support then look for k-itemsets and (k-1)-itemsets jointly for next (k-1) maximal itemsets and so on until no itemsets count found greater than minimum support..

Step 3: The maximal frequent transaction are found, than according to Apriori property consider each and every one its non empty subsets are frequent.

Step 4: There may be itemsets left over which are not included in maximal frequent itemset but they are frequent. Consequently find all frequent 1-itemset and reduce the database just consider only those transactions which contain frequent 1-itemset element but not contain the maximal frequent transaction.

Step 5: If no such transaction found then return otherwise go to step 6.

Step 6: Call SortRecursiveMine(ReducedDatabase) Procedure.

Output : Reduced Database and All frequent itemset

4. EXAMPLE

Suppose Table 1 is transactional database with Transactional Identity (TID), List of items and item count in each transaction. There are 9 transactions. Suppose the minimum support is 2.

Table 1 : Transactional Database , D

TID	List of item_IDs	Item Count
T001	MILK, BREAD, TOAST, SUGAR	4
T002	BREAD, JAM	2
T003	BREAD, BUTTER, TEA	3
T004	MILK, BREAD, JAM	3
T005	MILK, BUTTER	2
T006	BREAD, BUTTER	2
T007	MILK, BUTTER	2
T008	MILK, BREAD, BUTTER, TOAST	4
T009	MILK, BREAD, BUTTER	3

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

Table 2 : Candidate items, C1

Item Set	Support Count
{MILK}	6
{BREAD}	7
{BUTTER}	6
{JAM}	2
{TOAST}	2
{SUGAR}	1
{TEA}	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, L1

Item Set	Support Count
{MILK}	6
{BREAD}	7
{BUTTER}	6
{JAM}	2
{TOAST}	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
T008	**MILK, BREAD, BUTTER, TOAST**	**4**
T001	**MILK, BREAD, TOAST ,SUGAR**	**4**
T004	**MILK, BREAD, JAM**	**3**
T009	**MILK, BREAD, BUTTER**	**3**
T002	**BREAD, JAM**	**2**
T003	**BREAD, BUTTER**	**3**
T005	**MILK, BUTTER**	**2**
T006	**BREAD, BUTTER**	**2**
T007	**MILK, BUTTER**	**2**

Take 2-dimensional array; put the transaction into 2-dimensional array with their count of repetition.

2 Item set	Count	3 item set	Count	4 item set	Count
{MILK, BUTTER}	**2**	**{MILK, BREAD, TOAST}**	**1**	**{MILK, BREAD, BUTTER , TOAST}**	**1**
{BREAD, BUTTER}	**2**	**{MILK, BREAD, JAM}**	**1**		
{BREAD, JAM}	**1**	**{MILK, BREAD, BUTTER }**	**1**		

Find the maximal 4-itemset its count is 1. In our case, which is less then to given minimum support therefore this transaction is not considered as maximal frequent therefore now we scan 3-itemsets and 4-itemsets in array for maximal 3-itemsets jointly. This will result two maximal 3-itemsets

{MILK, BREAD, BUTTER} {MILK, BREAD, TOAST}

According to Apriori property subset of maximal frequent itemsets is also considered as frequent all subsets are

{MILK, BREAD, BUTTER} {MILK, BREAD, TOAST} {BREAD, BUTTER} {BREAD, TOAST} {MILK, BREAD}{MILK, BUTTER} {MILK, TOAST} {MILK} {BREAD}{BUTTER} {TOAST}.

Find the frequent 1-itemset from database it is found that {JAM} which is frequent but not include in maximal itemsets

There are itemsets remaining which are not included in maximal frequent itemset but they are frequent. Therefore find all frequent 1-itemset and reduce the database just consider only those transactions which contain frequent 1-itemset element but not include the maximal frequent transaction.

Reduce the database by considering only transaction which contains {JAM} itemset

TID	List of item_IDs
T002	**BREAD, JAM**
T004	**MILK, BREAD, JAM**

Go to next step because we have reduce database

By calling SortRecursiveMine (RediucedDatabase)

After scanning prune put items in 2-dimention array with the count of repetition

2 Item set	Count	3 item set	Count
{BREAD, JAM}	**1**	**{MILK, BREAD, JAM}**	**1**

Find the maximal 3-itemset its count is 1. In our case, which is less then to given support therefore this transaction is not considered as maximal frequent therefore now we scan 2-itemsets and 3-itemsets in array for maximal 2-itemsets jointly. This will result one maximal 2-itemsets

{BREAD, JAM}

Then final result of the frequent itemsets ({MILK, BREAD, TOAST} {BREAD, BUTTER, TOAST} {BREAD, BUTTER} {BREAD, JAM} {BREAD, TOAST} {MILK, BREAD} {MILK, BUTTER} {MILK, TOAST} {MILK} {BREAD} {BUTTER} {JAM}{TOAST})

5. EVOLUTION OF SORTRECURSIVE MINE ALGORITHM

As from reviewing the various techniques i.e. Apriori, FP-Growth, SortRecursive Mine and many more, we are trying to proposed differentiate them by the following considerations:

Table 5. Comparison Of Apriori , Fp-Growth And Sortrecursive Mine Algorithms

Algorithm Parameter	Apriori Algorithm	FP-Growth Algorithm	SR Mine Algorithm
Approach	Generate and test method	Divide and conquer method	Recursive method

Technique	It constructs an iterative approach, where k-itemsets are used to explore (k+1)-itemsets.	It constructs conditional frequent pattern tree and conditional pattern base from database which satisfy the minimum support.	It follows recursive approach to find maximal frequent itemsets that generates frequent subsets.
Memory utilization	enormous memory space required for candidate set in large database.	Low as for large database complete Tree structure cannot fit into main memory	little memory space required for two dimensional array in large database.
Database	Good for dense databases	Good for dense databases	Good for dense as well as for Sparse databases.
Scanning	Large scanning of database	Low scanning of database	Limited scanning of database as compare to Apripri algorithm.

6. PROS & CONS OF ALGORITHM

This type of approach is much better than Apriori an FP-Tree in terms of scanning and memory utilization because it does not produce larger number of candidates and also does not need to scan whole database again and again.

7. CONCLUSION

In this research work attempt has been made to develop an algorithm which is improvement over Apriori using an approach of improved SortRecursiveMine 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 recursion based approach. We have explained this new algorithm and illustrated with examples and try to compare with other methods. In our future work, we will try to implement, compare and various complexity of existing pattern mining algorithms.

8. 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

[10] J.R.Jeba,Dr S.P.Victor, "A Novel Approach for finding Frequent Item Sets with Hybrid Strategies",International Journal of Computer Applications (0975 – 8887) Volume 17– No.5, March 2011.

[11] Bharat Gupta,Dr.Deepak Garg,Karun Verma, "A Novel Approach to Mine Frequent Item Sets Using Maximal Apriori and FP-Tree Method",International Journal of Advance Computing Volume 3,Issue 2,April 2011.