A New Method to Mine Frequent Item Sets using Frequent Itemset Tree

Mr. Hardik S. Patel[1] Prof. Jigar N. Patel[2]
[1]P.G. Student [2]Internal Guide
[1, 2] Alpha College of Engg. & Technology, Khatraj, Kalol.

Abstract— The analysis of observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner. To find the association rules among the transactional dataset is the main problem of frequent itemset mining. Many techniques have been developed to increase the efficiency of mining frequent itemsets. In this paper, we denote a new method for generating frequent itemsets using frequent itemset tree (FI-tree). Also we describe the example of new method and its result analysis using wine dataset. Our method execution time is better compare to SaM method.

Keywords: Itemsets, Mine, Tree, SaM.

I. INTRODUCTION

The idea is to seek for something called knowledge, which means regularities, rule and structure hidden in the data. This activity is a subfield of computer science called knowledge discovery or sometimes data mining. This knowledge will help in making decisions and conclusions that lead to value creation for both the user and the owner of the data. For instance, the purchase information collected by a supermarket chain may help the supermarket to adjust product offering and availability to better suit the needs of its customers. A credit card company familiar with the purchase history of its customers can detect when a credit card has been stolen and used to buy goods or services that the customer would be unlikely to buy himself. Using location tracking technologies, a cell phone service provider can offer map-based services such as navigation or search of nearby restaurants. Moreover, bus and train companies can use recorded passenger data to help plan bus services to run more often where needed.

Apriori algorithm is quite successful for market based analysis in which transactions are large but frequent items generated is small in number. Apriori algorithm is used as a recommendation engine in an E-commerce system. Based on each visitor‘s purchase history the system recommends related, potentially interesting, products. It is also used as basis for a CRM system as it allows the company itself to follow-up on customer‘s purchases and to recommend other products by e-mail. To analyze the huge amount of data thereby exploiting the consumer behavior and make the correct decision leading to competitive edge over rivals[1]. Also Sequential association rule mining is one of the possible methods to analysis of data used by frequent itemsets[2].

A dataset is a set D of observations made over a set of attributes A More specifically, each observation in D is a Vector of (measured) values observed simultaneously for the set of attributes A. As a whole, D can be viewed as a matrix of n rows of observation vectors and m columns with attributes as headers. An observation can also be referred to as a data point. We denote a single attribute by a capital letter from the beginning of the alphabet, that is, by A, B, C . . ., and so on. An observation is denoted by the row vector t. Given an attribute A and a row t, we denote the value of attribute A on row t by t(A). Depending on the type of attribute A, the value of t(A) may be either

BINARY, THAT IS $t(A) \in \{0, 1\}$, OR NUMERICAL, THAT IS, $t(A) \in R$.

Binary valued attributes are often called items. Items are attributes that can be either present or absent at the moment of observation. the entire set of items is denoted by i. for datasets consisting of only binary attributes we have i = a, and by convention, a row t may be expressed as a subset of the universe of all attributes, i.e., $t \subseteq i$, containing those binary attributes

In this paper, we denote the overview of frequent itemset mining algorithms. Next, we denote a new method for generating frequent itemsets using frequent itemset tree (FI- tree). Also we describe the example of new method and its result analysis. Finally, conclude and future scope of the paper.

A. *Define a Problem:*

Mining of frequent itemset is acknowledged in the data mining field because of its broad applications in mining association rules, correlations, and graph pattern constraint based on frequent patterns, sequential patterns, and many other data mining tasks. Efficient algorithms for mining

Frequent itemsets are crucial for mining association rules as well as for many other data mining tasks. The major challenge found in frequent pattern mining is a large number of result patterns. As the minimum threshold becomes lower, an exponentially large number of itemsets are generated. Therefore, pruning unimportant patterns can be done effectively in mining process and that becomes one of the main topics in frequent pattern mining. Consequently, the main aim is to optimize the process of finding patterns which should be efficient, scalable and can detect the important patterns which can be used in various ways[3].

B. *Gap Analysis:*

All the algorithms produce frequent itemsets on the basis of minimum support. Apriori algorithm is quite successful for market based analysis in which transactions are large but frequent items generated is small in number[4]. Vertical Layout based algorithms claims to be faster than Apriori

but require larger memory space then horizontal layout based because they needs to load candidate, database and TID list in main memory[5]. For FP-Tree[6] and H-mine[7], performs better than all discussed above algorithms because of no generation of candidate sets but the pointes needed to store in memory require large memory space. For PASCAL algorithm finds both frequent and closed sets and it is 10 times as fast as Apriori but is only practical when the pattern length is short[8]. For FIAST algorithm, tries to reduce I/O, space and time but performance decreases for sparse datasets[9]. For SaM algorithm claims to be faster than all discussed above algorithms but require preprocessing on database results in execution time overhead[10]. Therefore these algorithms are not sufficient for mining the frequent itemsets for large transactional database

C. *Method of Frequent Itemset Mining:*

In a large transactional database like retailer database it is common that multiple items are selling or purchasing simultaneously therefore the database surly contains various transactions which contain same set of items. Thus by taking advantage of these transactions trying to find out the frequent itemsets and prune off the candidate itemsets whose node count is lower than min support using their FI-tree data structure without multiple database scan, results in efficiently execution time.

The method is used FI-tree, there are nodes holding frequent itemsets and transactions containing related itemsets. The principles of intersection and union are related to FI-tree. These principles are related to Frequent Itemset tree. In Frequent Itemset tree, there are nodes holding frequent itemsets and Tid containing related itemsets. The presented new method for mining frequent itemsets is a bottom-up level wise method that utilizes both Item set space and transaction space.

In order to construct k-itemsets, frequent (k-1)-itemsets are used. Their union is formed and for their support count and intersection operation is employed between the Tids of the itemsets.

Itemset {A} is in transactions with Tid 1, 4, 5, 8 and {B} is in transactions with Tid 1, 2, 4, 5, 8, i.e. T(A)={1, 4, 5, 8} and T(B)={1, 2, 4, 5, 8}. The Itemset {A, B} is the union of these two itemsets and intersection principles is used to find the tids for {A, B} as follows:

$T(AB) = T(A) \cap T(B) = \{1, 4, 5, 8\} \cap \{1, 2, 4, 5, 8\} = \{1, 4, 5, 8\}$

If the result is greater than minimum support, it will be joined to frequent Itemset tree. If the result is lower than minimum support, it will be pruned off. The more detail steps as follows:

Database D, minimum support are input by user and some or all frequent itemsets are output after execution of method. Assume that the frequent 1-itemsets and transaction sets, require no more memory that available and also there is space for generating candidate 2-itemsets from frequent 1-itemset. Scan database and find frequent 1-itemsets, at the same time obtain transaction sets, which includes the Itemset. Generate candidate 2-itemsets from frequent 1-itemset only. Prune off the candidate 2-itemsets whose node count is lower than min support using their Tidset. Now Frequent Itemset tree contains only frequent 2-itemsets at the second level. Consequently, for each frequent 3, 4,,,, n– itemset, based on node count to approve the consistence of the Itemset.

Now Frequent Itemset tree contains frequent 3 - itemsets, frequent 4-itemsets.... frequent n-itemsets at the third level, forth level.... n^{th} level respectively.

1) Example:

An example based on the database of retailer, D (table 1). There are six transactions in this database, that is, |D|=6. Suppose the minimum support is 3. We use the proposed new method for finding frequent itemsets in D, based on the intersection and union operation using below sample retailer transactional database (Table 1).

Tid	Items
10	1, 2, 3, 4
20	2, 3, 5, 6, 8
30	1, 2, 3, 5, 7
40	2, 5, 8
50	1, 3, 7, 8
60	2, 3, 7

Table (1): Sample retailer transactional database

Database D, minimum support = 3 are input by user and some frequent itemsets ({1}, {2}, {3}, {5}, {7}, {8}, {1, 3}, {1, 7}, {2, 3}, {2, 5}, {3, 7}, {1, 3, 7}).

After scanning a database put frequent 1-itemsets (table 2) with the count of repetition and Tid containing related itemsets. It is found that {4} and {6} itemsets, which is not frequent 1-itemset and removed. Generate candidate 2-itemsets from frequent 1- itemset only. Prune off the candidate 2-itemsets whose node count is lower than min support using their Tidset. Now FI-tree contains only frequent 2-itemsets at the second level and shown

(Table 3) Find frequent 3-Itemset based on node count to approve the consistence of the Itemset. Now FI-tree contains frequent 3-itemsets at the third level. Also frequent 3-itemsets shown (table 4).

Table–2
Frequent 1-Itemsets in array of Table – 1

Itemset	Frequency	Tid
{1}	3	10, 30, 50
{2}	4	20, 30, 40, 60
{3}	5	10, 20, 30, 50, 60
{5}	3	20, 30, 40
{7}	4	10, 30, 50, 60
{8}	3	20, 40, 50

Table–3
Frequent 2-Itemsets from Table – 2

Itemset	Frequency	Tid
{1, 3}	3	10, 30, 50
{1, 7}	3	10, 30, 50
{2, 3}	3	20, 30, 60
{2, 5}	3	20, 30, 40
{3, 7}	4	10, 30, 50, 60

Table–4
Frequent 3-Itemsets from Table – 3

Itemset	Frequency	Tid
{1, 3, 7}	3	10, 30, 50

II. RESULT ANALYSIS

In our experiments, we select wine dataset with different properties to prove the efficiency of the method. In wine dataset, 178 numbers of records and 14 numbers of columns (table 5). Wine dataset are used to test the new method by different settings of support thresholds. The SaM and new method are executed over the wine dataset. The total execution time taken for executing the SaM this method used wine dataset.

Files	Number of Records	Number of Columns
wine.data.txt	178	14

Table (5): Characteristics of dataset

Support (in %)	Total Execution time in second	
	SaM	New Method
30	3.70	3.26
45	1.89	1.82
60	0.22	0.18

Table (6): SaM and New Method Execution Time using wine dataset

Method (table 6) The total execution time for the new and SaM methods large reduces with the increase in support threshold from 30% to 60% for wine dataset. At 60% support threshold, two methods nearly matches the execution time. Our method takes less time as that compared to SaM method (figure 1).

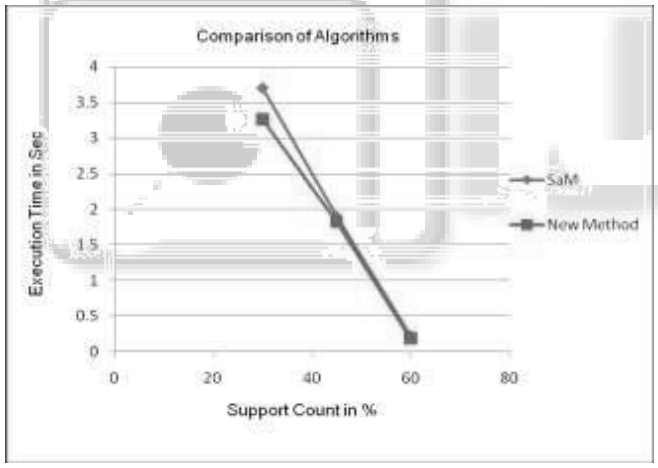

Fig (1): Execution time of wine dataset

III. CONCLUSION

By analytical study of the classical frequent itemset mining algorithms such as, Apriori, Eclat, FP-growth, Pascal, H-mine, Frequent Itemsets Algorithm for Similar Transactions (FIAST) and Split and Merge (SaM) To find out the advantages and disadvantages of these algorithms using different parameters. The SaM method is better from all above exiting methods.

We denote our new method for generating frequent itemsets by using frequent itemset tree (FI-tree). The analysis of total execution time for generating frequent itemsets denoted with standard dataset wine. Our method execution time is better compare to SaM method. At 60% support threshold, two methods nearly match the execution time.

We are use some constraints by user input for reduce total execution time for mining frequent itemsets. We are developing application based on this method.

REFERENCES

[1] Raorane A.A., Kulkarni R.V. and Jitkar B.D., Association Rule – Extracting Knowledge Using Market Basket Analysis, Res. J. Recent Sci.,1(2), 19-27 (2012)
[2] Shrivastava Neeraj and Lodhi Singh Swati, Overview of Non-redundant Association Rule Mining, Res. J. Recent Sci., 1(2), 108-112 (2012)
[3] Pramod S., Vyas O.P., Survey on Frequent Item set Mining Algorithms, In Proc. International Journal of Computer Applications, 1(15), 86–91 (2010)
[4] Agrawal R. and Srikant R., Fast algorithms for mining association rules, In Proc. Int'l Conf. Very Large Data Bases (VLDB), 487–499 (1994)
[5] Borgelt C., Efficient Implementations of Apriori and Eclat, In Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations (2003)
[6] Han J., Pei H. and Yin. Y., Mining Frequent Patterns without Candidate Generation, In Proc. Conf. on the Management of Data (2000)
[7] Yves Bastide, Rafik Taouil, Nicolas Pasquier, Gerd Stumme, Lotfi Lakhal, Mining Frequent Patterns with Counting Inference, In Proc.ACM SIGKDD, 66-75 (2000)
[8] Pei. J., Han. J., Lu. H., Nishio. S., Tang. S. and Yang. D., H-mine: Hyper-structure mining of frequent patterns in large databases, In Proc. Int'l Conf. Data Mining (2001)
[9] Duemong F., Preechaveerakul L. and Vanichayobon S., FIAST: A Novel Algorithm for Mining Frequent Itemsets, In Proc. Int'l Conf. Future Computer and Communication,140-144 (2009)10.
[10] Borgelt C., SaM: Simple Algorithms for Frequent Item Set Mining, IFSA/EUSFLAT 2009 conference (2009)

