© 2017 IJSRSET | Volume 3 | Issue 1 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

An Efficient Approach to Mine Frequent Itemsets Using the Variant of Classic Apriori and FP-Tree

Md. Towhidul Islam Robin[1], Ahmed Abdal Shafi Rasel[2] , Aiasha Siddika[3]
Stamford University Bangladesh, Dhaka, Bangladesh

ABSTRACT

As with the advancement of the information technologies, the amount of accumulated data is also increasing. It has resulted in large amount of data stored in databases, warehouses and other repositories. Thus the Data mining comes into picture to explore and analyse the databases to extract the interesting and previously unknown patterns and rules known as association rule mining. In data mining, association rule mining becomes one of the important tasks of descriptive technique which can be defined as discovering meaningful patterns from large collection of data. Mining frequent itemset is very fundamental part of association rule mining. Many algorithms have been proposed from last many decades including horizontal layout based techniques, vertical layout based techniques, and projected layout based techniques. But most of the techniques suffer from repeated database scan, Candidate generation (Apriori Algorithms), memory consumption problem (FP-tree Algorithms) and many more for mining frequent patterns. As in retailer industry many transactional databases contain same set of transactions many times, to apply this thought, in this paper we present a new technique which is combination of present Apriori (improved Apriori) and FP-tree techniques that guarantee the better performance in terms of time and memory than classical aprioi algorithm.
Keywords: Frequent Itemset, Association Rule Mining, FP-Tree, Apriori, Close Pattern, Cluster Based Mining.

I. INTRODUCTION

Data mining addresses two basic tasks: verification and discovery. The verification task seeks to verify user's hypotheses. While the discovery task searches for unknown knowledge hidden in the data. In general, discovery task can be further divided into two categories, which are descriptive data mining and predicative data mining. Descriptive data mining describes the data set in a summery manner and presents interesting general properties of the data. Predictive data mining constructs one or more models to be later used for predicting the behaviour of future data sets. There are a number of algorithmic techniques available for each data mining tasks, with features that must be weighed against data characteristics and additional business requirements. Among all the techniques, in this research, we are focusing on the association rules mining technique which is descriptive mining technique, with transactional database system. This technique was formulated by [2] and is often referred to as market basket analysis.

Association rules are one of the major techniques of data mining. Association rule mining finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories [13]. The volume of data is increasing dramatically as the data generated by day-to-day activities. Therefore, mining association rules from massive amount of data in the database is interested for many industries which can help in many business decision making processes, such as cross-marketing, Basket data analysis, and promotion assortment. The techniques for discovering association rules from the data have traditionally focused on identifying relationships between items telling some aspect of human behaviour, usually buying behaviour for determining items that customers buy together. All rules of this type describe a particular local pattern. The group of association rules can be easily interpreted and communicated.

A lot of studies have been done in the area of association rules mining. First introduced the

association rules mining in many studies have been conducted to address various conceptual, implementation, and application issues relating to the association rules mining task. Researcher in application issues focuses on applying association rules to a variety of application domains. For example: Relational Databases, Data Warehouses, Transactional Databases, and Advanced Database Systems (Object-Relational, Spatial and Temporal, Time-Series, Multimedia, Text, Heterogeneous, Legacy, Distributed [14].

Define the problem of finding the association rules from databases introduces of frequent pattern mining for discovery of interesting associations and correlations between itemsets in transactional and relational database. Association rule mining can be defined formally as follows:

I= {i1, i2, i3, …, in} is a set of items, such as products like (computer, CD, printer, papers, …and so on). Let DB be a set of database transactions where each transaction T is a set of items such that T⊆I. Each transaction is associated with unique identifier, transaction identifier (TID). Let X, Y be a set of items, an association rule has the form antecedent and is called the consequent of the rule where, set of items is called as an itemset or a pattern[7]**.** Let n be the number of rows (transactions) containing itemset in the given database. Frequent patterns, such as frequent itemsets, substructures, sequences term-sets, phrase sets, and sub graphs, generally exist in real-world databases. Identifying frequent itemsets is one of the most important issues faced by the knowledge discovery and data mining community. Frequent itemset mining plays an important role in several data mining fields as association rules [1] warehousing [9], correlations, clustering of high-dimensional biological data, and classification [13]. Given a data set d that contains k items, the number of itemsets that could be generated is 2k - 1, excluding the empty set[1]. In order to searching the frequent itemsets, the support of each itemset must be computed by scanning each transaction in the dataset. A brute force approach for doing this will be computationally expensive due to the exponential number of itemsets whose support counts must be determined. There have been a lot of excellent algorithms developed for extracting frequent itemsets in very large databases. strategies adopted by these algorithms: the first is an effective pruning strategy to reduce the combinatorial search space of candidate itemsets (Apriori techniques). The second strategy is to use a compressed data representation to facilitate in-core processing of the itemsets (FP-tree techniques). Database has been used in business management, government administration, scientific and engineering data management and many other important applications. The newly extracted information or knowledge may be applied to information management, query processing, process control, decision making and many other useful applications. With the explosive growth of data, mining information and knowledge from large databases has become one of the major challenges for data management and mining community.

The frequent itemset mining is motivated by problems such as market basket analysis [3]. A tuple in a market basket database is a set of items purchased by customer in a transaction. An association rule mined from market basket database states that if some items are purchased in transaction, then it is likely that some other items are purchased as well. Finding all such rules is valuable for guiding future sales promotions and store layout. The problem of mining frequent itemsets are essentially, to discover all rules, from the given transactional database D that have support greater than or equal to the user specified minimum support.Counting of resolutions of a constraint tree is an expensive procedure. But if the number of constraints, k is small, the tree construction is quite feasible. In practice, k is typically small, and in our case k=3, so this algorithm will be of practical use.

II. METHODS AND MATERIAL

A. Problem Formulation

Let I = { i1, i2,…….n }be a set of items and n is considered the dimensionality of the problem. Let D be the task relevant database which consists of transactions where each transaction T is set of items such that T £ I. A transaction T is said to contain itemset X, which is called a pattern. A transaction T is said to be maximal frequent if its pattern length is greater than or equal to all other existing transactional patterns and also count of occurrence (support) in database is greater than or equal to specified minimum support threshold [15]. An itemset X is said to be frequent if its support is greater than or equal to give the possible minimum support threshold.

Transactional database D and minimum support

threshold is given, therefore the problem is to find the complete set of frequent itemsets from Transactional type of databases to increase the business, so that relation between customers behaviour can be found.

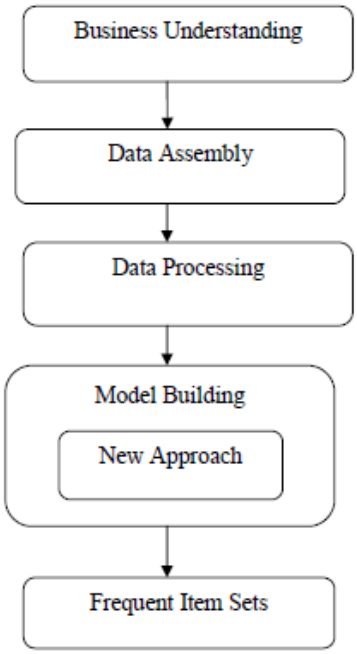

Figure 1: Methodology used to find frequent patterns.

FP-tree algorithm [5, 6] is based upon the recursively divide and conquers strategy; first the set of frequent 1-itemset and their counts is discovered. With start from each frequent pattern, construct the conditional pattern base, then its conditional FP-tree is constructed (which is a prefix tree.). Until the resulting FP-tree is empty, or contains only one single path. (Single path will generate all the combinations of its sub-paths, each of which is a frequent pattern). The items in each transaction are processed in L order. (i.e. items in the set were sorted based on their frequencies in the descending order to form a list).

B. Construction of FP-Tree

Create root of the tree as a "null". After scanning the database D for finding the 1-itemset then process the each transaction in decreasing order of their frequency. A new branch is created for each transaction with the corresponding support. If same node is encountered in another transaction, just increment the support count by 1 of the common node. Each item points to the occurrence in the tree using the chain of node-link by maintaining the header table. After above process mining of the FP-tree will be done by Creating Conditional (sub) pattern base[12]. Start from node constructs its conditional pattern base. Then, Construct its conditional FP-tree & perform mining on such a tree. Join the suffix patterns with a frequent pattern generated from a conditional GP-tree for achieving FP-growth. The union of all frequent patterns found by above step gives the required frequent itemset can be found in [10]. The problem of mining frequent itemsets arises in the large transactional databases when there is need to find the association rules among the transactional data for the growth of business. Many different algorithms has been proposed and developed to increase the efficiency of mining frequent itemsets including (Horizontal layout based algorithms, Vertical Layout Based algorithms [1, 2, 4, 8, 9, 10], Projected layout based algorithms and Hybrid algorithms. There are several ways to mine the frequent patterns using horizontal layout based approach. FP-tree is one of the prominent approach to find this itemset which satisfies the minimum support which must be given initially by the system and minimum confidence is one the important parameter to prune the huge set of itemsets. The FP-tree built for the conditional pattern base X is called conditional FP-tree. Let sample database in table 1 where Tid denotes unique transaction id and I1, I2, I3 indicates number of itemsets of a particular transaction.

Table 1: Sample transactions with itemsets

Tid	Items
T1	I1, I2, I3, I4, I5
T2	I5, I4, I6, I7, I3
T3	I4, I3, I7, I1, I8
T4	I4, I7, I9, I1, I10
T5	I1, I5, I10, I11, I12
T6	I1, I4, I13, I14, I2
T7	I1, I4, I6, I15, I2
T8	I16, I7, I9, I17, I5
T9	I1, I9, I8, I10, I11
T10	I4, I9, I12, I2, I14
T11	I1, I3, I5, I6, I15
T12	I3, I7, I5, I17, I16
T13	I8, I3, I4, I2, I11
T14	I4, I9, I13, I12, I18
T15	I5, I3, I7, I9, I15
T16	I18, I7, I5, I1, I3
T17	I1, I17, I7, I9, I4
T18	I4, I3, I16, I5, I1

Table 2: Support Count Result

Item	Support	Item	Support
I1	11	I10	3
I2	4	I11	3
I3	9	I12	3
I4	10	I13	2
I5	10	I14	2
I6	3	I15	3
I7	8	I16	3
I8	3	I17	3
I9	7	I18	3

Suppose minimum support is 5. Thus delete all infrequent items whose support is less than 5.After all the remaining transactions arranged in descending order of their frequency. Create a FP- tree[4]. For Each Transaction create a node of an items whose support is greater than minimum support, as same node encounter just increment the support count by 1.

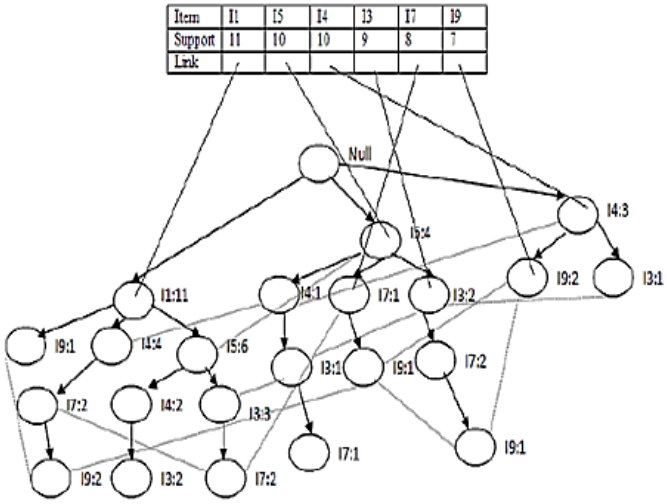

Figure 2: FP-Tree Constructed for Sample Database.

C. Implementation of Proposed Approach

Data Assembly also include collecting of data, for testing purpose the data is collected from the random source. The data is challenging due to the number of characteristics which are the number of the records, and the sparseness of the data (each records contains only small portion of items). In our experiments we chose different dataset with different properties, to prove the efficiency of the algorithms, Table 3 shows the datasets and the characteristics such as length and type.

Table 3: Data set properties

Data set	#Items	Avg. Length	#Trans	Type	Size
T10I4D100K	1000	10	100, 000	Sparse	3. 93 MB
Mushroom	119	23	8, 124	Dense	557 KB

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 the database as early as possible without generating the candidate itemset and multiple database scan, results in efficiently usage of memory and improved computation. This proposed algorithm is based upon the Apriori property [2] i.e. all non-empty subsets of the frequent itemsets are frequent. Algorithm has two procedures. In first procedure, find all those maximal transactions which are repeating in the database equal to or greater than min user defined support also known as maximal frequent itemset [15]. Then get all nonempty subsets of those maximal frequent itemset as frequent according to Apriori property. Scan the database to find 1-itemset frequent elements. There may be many items found which are 1-itemset frequent but not include in maximal frequent transactions. Therefore prune the database by just considering only those transactions from the database which contain 1-itemset frequent elements, but not include in the maximal frequent itemsets. Now this pruned database is smaller than the actual database in the average cases and no item left in best case. For the second procedure, pruned database is taken as input and scan the pruned database once find 1-itemset frequent and delete those items from transaction which are not 1-itemset frequent. Then construct the FP-tree [6] only for pruned transactions. In this way it reduces the memory problem for FP-tree because the database is reduced in most of cases. In best case no need to build FP-tree because all elements are found in first procedure. In the worst case if there is no maximal frequent transaction exist, then only second procedure run and also computational performance is same as FP-tree[11]. The key of this idea to prune the database after finding the maximal frequent itemsets and formation of FP-tree for a pruned database thus reduce memory problem in FP-tree and make the mining process fast. The more detail step as follows:

Procedure1:

Input: Database D, minimum support

Step 1: Take a 2- dimensional array; Put the transaction into 2-dimmension array with their count of repetition.

Step 2: Arrange them in increasing order on the basis of the pattern length of each transaction.

Step 3: 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-itemsets 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. If no such transaction found then go to Procedure2.

Step 4: Once the maximal frequent transactions found, than according to Apriori property consider all its non-empty subsets are frequent.

Output: some or all frequent itemsets, Pruned database-D1.

Procedure2:

Input: Pruned database D1, minimum support

Step 1: Find frequent 1-itemset from pruned database; delete all those items which are not 1-itemset frequent.

Step 2: Construct FP-tree for mine remaining frequent itemset by following the procedure of FP-tree algorithm as discussed above in section 2B.

Output: Remaining frequent itemsets.

2- itemset	count	3-itemset	count	4-itemset	count
{I2,I3}	2	{I1,I2, I4}	1		
{I1,I3}	2	{I1,I2, I3}	1	{I1,I2,I3, I5}	
{I2,I3}	2	{I1,I2, I4}	1		2

Table 4: (step-1) After scanning a database put items in 2-dimensional array with the count of repetition.

According to step-2 find maximal itemset (4-itemset). Check weather its count is greater or equal to specified support, its count is 2 in our case which is equal to given support therefore this transaction is considered as maximal frequent. (If its count is less than support value then we scan k-1 and k-itemset in array for k-1 maximal itemset jointly and so on until finding all maximal frequent itemset from a array. i.e. 3-itemset and 4-itemset for checking 3-itemset maximal and so on). According to Apriori property (step 3) subset of maximal frequent itemset is also considered as frequent .i.e. Maximal frequent itemset: {I1, I2, I3, I5}.All subsets are frequent (Apriori Property) i.e. {I1, I2, and I3},{I1, I2, I5}, {I2, I3, I5}, {I2, I3}, {I2, I5}, {I1, I2}, {I1, I3}, {I1, I5}, {I3, I5}, {I1}, {I2}, {I3}. While scan the database for finding the above mined support to find 1-itemset frequent from database, it is found that I4 which is frequent butnot include in maximal frequent itemset. (There may be many items remain which are not include in maximal frequent itemsets, in our case only 1 item is there). Prune the database by considering only transaction which contains I4 itemset.

Output: Some frequent itemsets ({I1, I2, I5}, {I2, I3, I5}, {I2, I3}, {I2, I5}, {I1, I2}, {I1, I3}, {I1, I5}, {I3, I5},{I1},{I2},{I3}).

III. RESULTS AND DISCUSSION

Time Comparison

As a result of the experimental study, revealed the performance of our new technique with the Apriori and FP-Growth algorithm. The run time is the time to mine the frequent itemsets. The experimental result of time is shown in Figure 3 reveals that the proposed scheme outperforms the FP-growth and the Apriori approach.

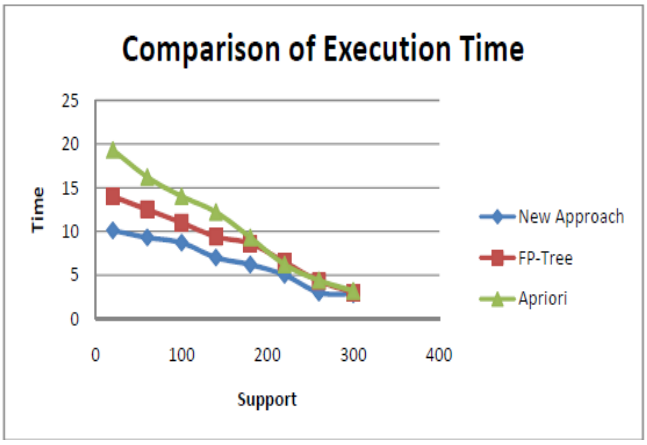

Figure 3: The execution time for mushroom dataset.

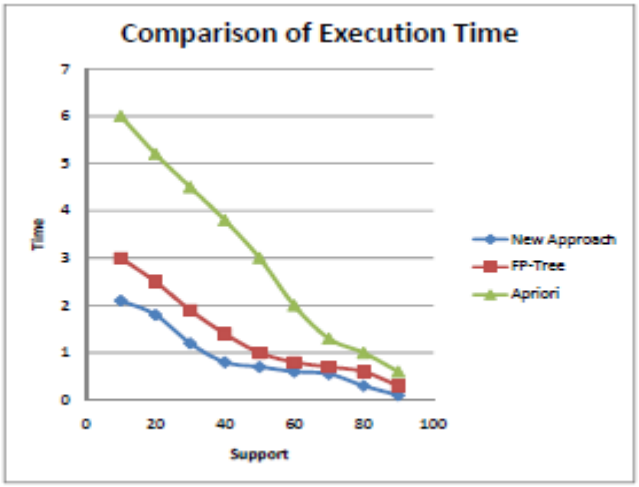

Figure 4 : The execution time for Artificial dataset

As it is clear from the comparison new algorithm performs well for the low support value for the mushroom dataset which contains 8124 transactions and average length of items 23. But at the higher support its performance matches the FP-Tree and Apriori algorithms. Apriori performs with larger time. FP-tree produces the approximately same execution as of new approach in later stages. For the artificial dataset which contains the maximal frequent itemset in large amount shows better result with new approach as shown in figure 3 then FP-tree and Apriori algorithm. In the artificial dataset there are various transactions consider which occur repeatedly in the database and some transactions occur greater than the minimum support. The itemset remains for mining frequent itemset are mined with the help of second procedure whose complexity equals to the FP-Growth algorithm but due to procedure 1 the overall complexity reduce and become efficient and more accurate than classic apriori.

Memory-Comparison

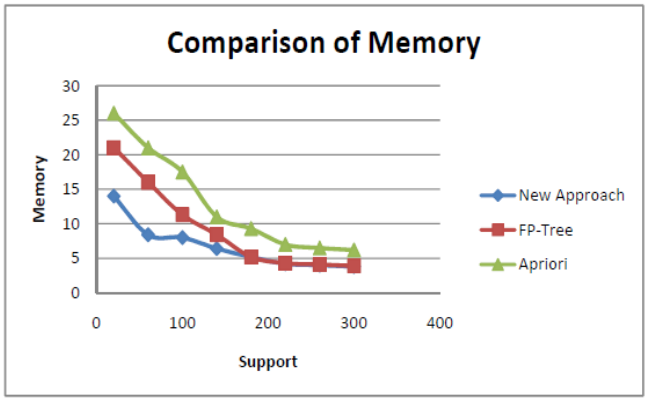

Figure 5 : The memory usage at various support levels on Mushroom dataset.

As it is clear from figure 5, the memory consumption for the Apriori algorithm is the highest at all level support because it produces candidate itemsets. The memory consumption for FP-tree at higher support levels is approximately same as the new approach because as the support increase the probability of finding the maximal itemset whose repetition is greater than the minimum support. is less thus its working become same as the FP-Growth algorithm

The below figure 6 shows the comparison takes place at sparse dataset, in sparse dataset data sets containing more enough zero entries (unmarked fields or items), in other words, the ratio number of fields / number of elements for the data database is smaller. Since the Apriori algorithm stores and processes only the non-zero entries, it takes the advantage of pruning most of the infrequent items during the first few passes. Therefore At high levels support the performances of our proposed scheme and Apriori are near. But at lower levels it shows that new approach performs well at all support level in consumption of memory. In this case also Apriori consume large amount of memory which is more than the FP-Tree and new approach due to its candidate generation problem. FP-tree approach performs better than the Apriori but not than the new approach.

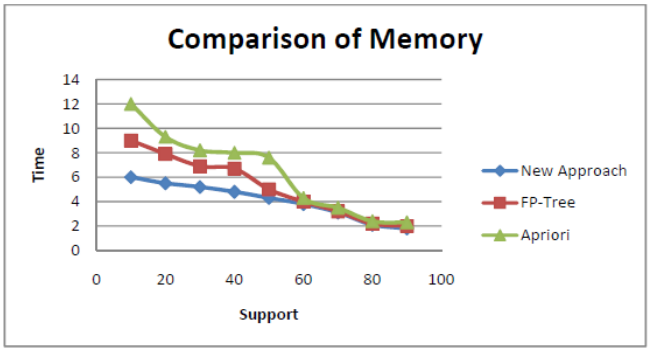

Figure 6: The memory usage at various support levels on Artificial dataset.

IV. CONCLUSION

We considered the following factors for creating our new scheme, which are the time and the memory consumption, these factors are affected by the approach for finding the frequent itemsets. Work has been done to develop an algorithm which is an improvement over Apriori and FP-tree with using an approach of improved Apriori and FP-Tree algorithm for a transactional database. According to our observations, the

performances of the algorithms are strongly depends on the support levels and the features of the data sets (the nature and the size of the data sets). Therefore we employed it in our scheme to guarantee the time saving and the memory in the case of sparse and dense data sets. It is found that for a transactional database where many transaction items are repeated many times as a super set in that type of database maximal Apriori (improvement over classical Apriori) is best suited for mining frequent itemsets. The itemsets which are not included in maximal super set is treated by FP-tree for finding the remaining frequent itemsets. Thus this algorithm produces frequent itemsets completely. This approach doesn't produce candidate itemsets and building FP-tree only for pruned database that fit into main memory easily. Thus it saves much time and space and considered as an efficient method as proved from the results.

V. REFERENCES

[1]. A. Savasere, E. Omiecinski, and S. Navathe. "An efficient algorithm for mining association rules in large databases". In Proc. Int'l Conf. Very Large Data Bases.

[2]. Aggrawal.R, Imielinski.t, Swami.A. "Mining Association Rules between Sets of Items in Large Databases". In Proc. Int'l Conf. of the 1993 ACM SIGMOD Conference Washington DC, USA.

[3]. Agrawal.R and Srikant.R. "Fast algorithms for mining association rules". In Proc. Int'l Conf. Very Large Data Bases (VLDB), Sept. 1994, pages 487–499.

[4]. Brin.S, Motwani. R, Ullman. J.D, and S. Tsur. "Dynamic itemset counting and implication rules for market basket analysis". In Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD), May 1997, pages 255–264.

[5]. C. Borgelt. "An Implementation of the FP- growth Algorithm". Proc. Workshop Open Software for Data Mining, 1–5.ACMPress, New York, NY, USA 2005.

[6]. Han.J, Pei.J, and Yin. Y. "Mining frequent patterns without candidate generation". In Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD),2000.

[7]. Park. J. S, M.S. Chen, P.S. Yu. "An effective hash-based algorithm for mining. association rules". In Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD), San Jose, CA, May 1995, pages 175–186.

[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 (ICDM),November 2001.

[9]. C.Borgelt. "Efficient Implementations of Apriori and Eclat". In Proc. 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations, CEUR Workshop Proceedings 90, Aachen, Germany 2003.

[10]. Toivonen.H. "Sampling large databases for association rules". In Proc. Int'l Conf. Very Large Data Bases (VLDB), Sept. 1996, Bombay, India, pages 134–145.

[11]. Nizar R.Mabrouken, C.I.Ezeife. Taxonomy of Sequential Pattern Mining Algorithm". In Proc. in ACM Computing Surveys, Vol 43, No 1, Article 3, November,2010.

[12]. Yiwu Xie, Yutong Li, Chunli Wang, Mingyu Lu. "The Optimization and Improvement of the Apriori Algorithm". In Proc. Int'l Workshop on Education Technology and Training & International Workshop on Geoscience and Remote Sensing 2008.

[13]. "Data mining Concepts and Techniques" by By Jiawei Han, Micheline Kamber, Morgan Kaufmann Publishers, 2006.

[14]. S.P Latha, DR. N.Ramaraj. "Algorithm for Efficient Data Mining". In Proc. Int'l Conf. on IEEE International Computational Intelligence and Multimedia Applications, 2007, pp. 66-70.

[15]. Dongme Sun, Shaohua Teng, Wei Zhang, Haibin Zhu. "An Algorithm to Improve the Effectiveness of Apriori". In Proc. Int'l Conf. on 6th IEEE Int. Conf. on Cognitive Informatics (ICCI'07), 2007.