Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 45 (2015) 101 – 110

Procedia Computer Science

International Conference on Advanced Computing Technologies and Applications (ICACTA-2015)

An optimized algorithm for association rule mining using FP tree

Meera Narvekar[a], Shafaque Fatma Syed[b]

[a]Computer Department, DJ Sanghvi College of Engineering, Mumbai and 400056, India
[b]Computer Department, DJ Sanghvi College of Engineering, Mumbai and 400056, India

Abstract

Data mining is used to deal with the huge size of the data stored in the database to extract the desired information and knowledge. It has various techniques for the extraction of data; association rule mining is the most effective data mining technique among them. It discovers hidden or desired pattern from large amount of data. Among the existing techniques the frequent pattern growth (FP growth) algorithm is the most efficient algorithm in finding out the desired association rules. It scans the database only twice for the processing. The issue with the FP growth algorithm is that it generates a huge number of conditional FP trees. In the proposed work, we have designed a new technique which mines out all the frequent item sets without the generation of the conditional FP trees. Unlike FP tree it scans the database only once which reduces the time efficiency of the algorithm. It also finds out the frequency of the frequent item sets to find out the desired association rules.

Peer-review under responsibility of scientific committee of International Conference on Advanced Computing Technologies and Applications (ICACTA-2015).

*Keywords:*FP tree; association rules; frequent item sets; mining.

1. Introduction

Data mining is used to deal with very large amount of data which are stored in the data ware houses and databases, to find out desired interesting knowledge and information. Many data mining techniques have been proposed such as, association rules, decision trees, neural networks, etc. It has become the point of attention from many years. One of the most well-known techniques is association rule mining. It is the most efficient data mining technique. It discovers the hidden patterns from the large databases. It is responsible to find the relationship between the different attributes of data.

Association rules were first introduced in [1]. It provides the results in the form of "if-then" statements. These rules are generated from the input datasets. The rules are derived from the support and confidence value given as input from the user. An association rule is, in general, an expression of the form X→Y, where X is an antecedent and Y is

Peer-review under responsibility of scientific committee of International Conference on Advanced Computing Technologies and Applications (ICACTA-2015).
doi:10.1016/j.procs.2015.03.097

a consequent. Association rule shows how many times Y has occurred if X has already occurred depending on the support and confidence value. Many algorithms for generating association rules were presented over time. Some well-known algorithms are Apriori and FP-Growth.

Apriori algorithm is one of the most well-known algorithms for association rule mining. It uses a breadth-first search technique for counting the support of item sets and uses the candidate generation for getting the rules. It uses the downward closure property of the support and also uses bottom-up strategy. Apriori algorithm uses the candidate generation through which the frequent item sets are generated.

FP-growth (frequent pattern growth) uses a prefix-tree (FP-tree) structure to store the database in a compressed form. FP-growth adopts a divide-and-conquer strategy for finding the frequent item sets. It first compresses the database into FP tree which has the items and their associations then it divides the tree into smaller tress called as conditional FP trees and mines out the frequent item sets separately.

We propose an efficient association rule mining technique. The main advantage of this is we can get all the frequent item sets with fewer efforts. The proposed technique also scans the database only once. It generates the frequent item sets without generation of any conditional FP trees.

2. Literature Review

Association mining by using the Apriori algorithm gives good rules but when the database size increases its performance drops. This is because it has to scan the entire database every time it scans the transaction. Apriori algorithm uses a breadth first search technique to get the large item sets. The issue with this algorithm is that it cannot be applied directly to mine large databases. The other well-known algorithm is Frequent Pattern (FP) growth algorithm. It uses the divide-and-conquer technique. FP forms a tree of frequent patterns and computes the frequent item sets. When we compare FP growth and Apriori algorithm, FP is much more superior in case of efficiency. But the setback of the FP is that it produces a large number of conditional FP-Trees.

The author of [2] Yan Hu proposed an optimized algorithm to mine maximal frequent item sets. During the process of frequent item mining, we sometimes need to deal with large numbers of candidate item sets. However, since frequent item sets are upward closed, it is sufficient to discover only all maximal frequent item sets (MFI). A frequent item set is maximal if it has no superset that is frequent. The structure helps FP max reduce the number of subset checking and save the search time [2]. In this algorithm it is considered that for some conditional pattern bases that satisfy certain conditions, corresponding maximal frequent item sets can be acquired as early as possible. This will reduce consumption in following FP-tree construction and improve mining efficiency.

The author of [3] Li Juan, proposed QFP algorithm, through scanning the database only once, the QFP algorithm can convert a transaction database into a QFP tree after data preprocessing, and then do the association rule mining of the tree[3]. The QFP algorithm has more integrity than the FP growth algorithm, and retain the complete information for mining frequent patterns; it will not destroy the long pattern of any transaction, and significantly reduce the non-relevant information[3]. The input of the QFP algorithm is the same as that of the FP growth algorithm or the Apriori algorithm, therefore, the QFP algorithm may apply to any situation which is suitable for the FP growth algorithm or the Apriori algorithm [3].

The author of [4] Jun TAN, proposes an effective closed frequent pattern mining algorithm at the basis of frequent item sets mining algorithm FP-growth. A novel FP-array technology is proposed in the algorithm, a variation of the FP-tree data structure is used which is in combination with the FP-array. A closed frequent item set is applicable when a transaction database is very dense, i.e. when the database contains large number of long frequent item sets, mining all frequent item sets might not be a good idea. This algorithm works well in the databases which are sparse.

The author of [5] Syed Khairuzzaman Tanbeer has introduced the dynamic tree restructuring concept. The author has proposed CP-tree which, by dynamically applying a tree restructuring technique, achieves a frequency-descending prefix-tree structure with a single-pass and considerably reduces the mining time to discover frequent patterns from a dataset. In their work they have concluded that with relatively small tree restructuring overhead, CP-tree achieves a remarkable performance gain in terms of overall runtime. The easy maintenance and constant access to full database information in a highly compact fashion facilitate CP-tree's applicability in interactive, incremental, and other frequent pattern mining applications [5].

The author of [6] Ye Gao, has proposed an algorithm which contra poses the defects of FP-growth: traveling the same path repeatedly and failing to release nodes processed in time. It uses a strategy that ergodic the path only once, and the conditional patterns of all nodes are obtained by adding m_sFrearea and m_sFlag node domains. Meanwhile, it adopts the strategy of deleting FP-tree and releases the processed nodes to reduce the memory which the algorithm occupies, so it improves the efficiency of the algorithm [6].
The other algorithm described in [7-10] explains in the detail the FP growth algorithm. The author of [11] presents some fast algorithms for mining association rules. They proposed Apriori Tid algorithm, for discovering all significant association rules between items in a large database of transactions.

3. Concepts of association rule mining

The association rule mining was first introduced by [1]. An association rule can be defined as follows, let I= {i1,i2,i3,i4….} be a set of items. Let the database be a set of transactions where each transaction T is a subset of I. An association rule is an inference of the form $X \Rightarrow Y$, where X, Y is a subset of I and $X \cap Y = \varphi$. The set of items X is called antecedent and Y is called as the consequent. The two properties confidence and support are generally considered in association rule mining. Let freq(X) be the number of rows containing X item set in the given database. The support of the item set X is defined as the fraction of all rows containing the item set, i.e. freq(X)/D.

The support of an association rule is the support of the union of the antecedent X and the consequent Y, i.e.
support $(X \rightarrow Y) = (X \cup Y) / D$

The confidence of an association rule is defined as the percentage of rows in D containing item set X that also contain item set Y, i.e.

Confidence $(X \rightarrow Y) = P(X/Y) =$ support $(X \cup Y)$/support(X)

An item set is frequent if its support is equal to or more than a user specified minimum support. Association rule mining is to identify all rules meeting user specified constraints such as minimum support and minimum confidence.

4. Proposed methodology

The main idea of the paper is to design a technique for association rulemining which reduces the complex intermediate process of the frequent item set generation. Through this work we aim to design a technique for association rule mining to reduce the no. of times the transactional database is scanned, reduce the no. of conditional pattern bases generated, and remove the generation of the conditional FP trees.

There are three steps involved in the proposed technique. In the first step, we scan the database one time to generate a D-tree from the database. The D-tree is basically the replica of the database under consideration. After scanning the database and getting the D-tree out of it, we then use the D-tree for further processing. We then scan the D-tree to count the no. of occurrences of each item in the database and record it as frequency of each item. Now here to calculate the frequency we scan the tree and not the database, since reading a transaction from the memory-resident tree structure is faster than scanning it from the disk [5].

In the second step, using the D-tree and support count as an input we construct a New Improved FP tree and a node table which contains the some nodes and their frequencies the details of which will be given while discussing the algorithm. For the construction of new improved FP tree we use the Algorithm 1 mentioned in further sections.
In the third step, we take as input the New Improved FP tree along with the support, a count from the node table and the frequency of each item in FP tree and apply Algorithm 2 to get the frequent item sets.
Now let us see each step in detail. For the understanding the algorithm in detail let us consider an example. Table-1 shows the sample transactional database.

4.1. Construction of the D-tree

Consider the table-1 for the construction of the D-tree. To construct the D-tree we will consider one by a b f cone all the transactions of the database. We first will create a root node for the tree as null. Then for each transaction in the database we create a path in the D-tree from the root. Also every node will have a count which will identify that

how many times that node is shared. If some parts of the transactions are common in two given transactions then they may share some nodes in the tree. But the shared nodes will always have count greater than 1. Now with this idea we proceed to make the D-tree. Let us consider the first transaction a--b--c. For this transaction one path will be formed from the root node as a:1--b:1--c:1. Then similarly for the second transaction a new path from root would be formed as b:1--e:1. Now, for the third transaction b-f, we can see that node b already exists as the first node for the second transaction so it's the common node between these two transactions so it can be shared. Hence, for third transaction the count for the node b will be incremented by 1 and a new child node e:1 would be added to it. In the similar way we will construct the tree with the remaining transactions in the database. The fully constructed D-tree is shown in Fig.1. This completes the first step of the technique which is the construction of the D-tree.

After the construction of the D-tree, in the first step we also need to compute the frequencies of each item in the database. For achieving so, we need to scan the D-tree once to get individual count of each item. We can use any tree traversal algorithm for this purpose and keep track of each item's frequency at the end of scanning we would be having total count of each item. After this procedure the output for the considered sample transactional database is as given in table-2.

Table 1. Sample transactional database

T_id	Transaction
1	a b c
2	b e
3	b f
4	a b e
5	a f
6	b f
7	a f
8	a b f c
9	a b f

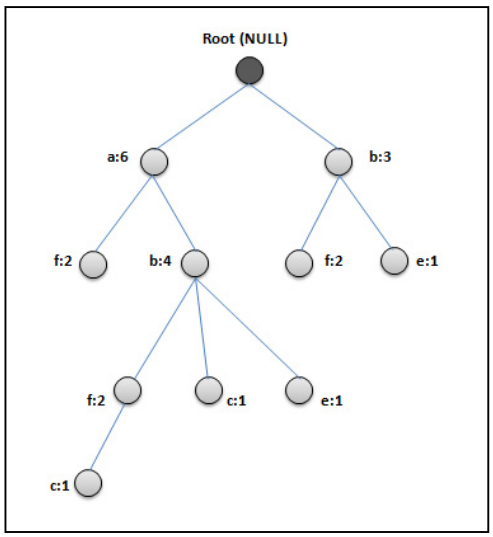

Fig. 1. Construction of D-tree.

Table 2. Item frequency.

Item	Frequency
b	7
a	6
f	6
c	2
e	2

4.2. Construction of the New Improved FP tree

In the second step of the proposed technique we apply Algorithm 1 and get the New Improved FP tree. The input to the algorithm is the D-tree and the support count and its output id the New Improved FP tree and a node table. The tree is used to more specifically represent the correlation among the items in the transactional database. The node table is used to store some items which are spare. The node table consists of two columns item_name and frequency. The item_name is the name of the item and frequency is the no. of occurrences of the item in the node table. The idea behind introducing the node table is that in the original FP tree there are a number of branches and the same item appears in more than one node. In our proposed technique every unique item has only one node in the tree. Hence, it is simple and efficient for processing. Now in order to put any item in node table there are certain cases to be considered. Under any given condition while making the tree if any case comes true we put the item in the node table. There are two cases under which we put any item in node table.

Case 1: When the item under consideration has no edge from the current considered root to its node. This means that the item is already present in the FP tree. The root keeps on changing throughout the course of making the tree, it's not fixed.

Case 2: If a transaction does not contain the most frequent item then we put all the items in the node table.
Let us now see the Algorithm-1. The detailed working of the algorithm is as follows:

Algorithm 1: Construction of the New Improved FP-tree
Input: D-Tree, support count
Output: New Improved FP-tree, frequency of each item in the New Improved FP-tree, and a count

Step 1: Create the root node R for the tree. Since, it is a prefix tree, R= NULL.
Step 2: For each considered path from the D-tree, do
Step 3: Sort the items in the considered path in descending order of the frequency, according to the number of occurrence in transaction.
Step 4: Let the path which is descending ordered represented by [k|L], where k is the first item and L represent the rest of the items in path.
Step 5: If k= most frequent item in the database, do the steps 6 and step7 else go to step 8.
Step 6: If the Root R has a direct child node N, such that N's name_of_item = k's name_of_item, Increase the count of the item N by 1. Transfer Root from R to k.
Step 7: For the each remaining item in L do the following steps.
[a] Create a new child node with count as 1 for each new item from current root, transfer the root to new node.
[b] If an item, Li has no single edge from the current root to its node, where Li is an item in L and already exists in new FP-tree. Then Li is an item to be put in table. Store Li with frequency 1 in table.
Step 8: For each item in the considered path [k|L]. Store all items in the table with frequency 1.
Step 9: Sum up the frequencies in the table if there is more than one entry for a node in the table. Along with the frequency output the New Improved FP-tree.

Since, most of the relations of our interest are related to the most frequent item. Hence, in the proposed technique, the maximum attention is focused on the most frequent item in the database. Hence, when we consider each path from the D-tree we arrange all the items in the descending order of the frequency so that the most frequent item always lies in the first position in the transaction and remains in the top level nodes in the New Improved FP tree. The root of the tree is changed at the step 6 of the algorithm, also at step 6 if an item is repeated its frequency is increased by 1. The root of the tree is changed so that the correlation is seen clearly among the items in the database. For putting the items in the node table the checking is performed at the step 5 of the algorithm. The Case 1 has been put up at the step 7[b] and Case 2 has been put up at the step 8 of the algorithm. Here we represent the FP tree frequency by F.

4.3. Mining Frequent Item Sets

The traditional FP growth algorithm needs to generate a large number of conditional pattern bases then calculate the conditional FP-tree, to avoid candidate generation. But in the case for larger databases it is not efficient and need a large memory requirement that can over stack the main memory [5]. We will use the algorithm 2 for the generation of the frequent item sets. The input of this algorithm is the new improved FP tree and the support count from the user and output of the frequent item sets. After the application of the Algorithm 2 the valid set of the association rules can be found out using the concepts given at Section III.

Algorithm 2: For Frequent Item Set Mining
Input: New Improved FP-tree, support-S, count-C, frequency-F
Output: Frequent Item Set

```
for each item q in New Improved FP-tree
do
{
   if (q.F<q.S)
   Generate the frequent item set as all the possible combinations of item considered and all intermediate nodes up
   till the most frequent item node in New Improved FP-tree, the frequency will be q.F+C
         else if (q.F = = q.S)
         Generate frequent item sets as all the possible combinations of the item considered and nodes having higher
         frequency in New Improved FP-tree, the frequency will be q.F
                  else
                  Generate frequent item set as all the possible combinations of item and its parent node in New
                  Improved FP-tree, the frequency will be q.F
}
```

After the application of the algorithm 2 we get the frequent item sets with their frequencies i.e. IF (Frequent item set frequency). We will us the frequent items set frequency (IF) to get the association rules from the frequent item sets. The frequent item set mining algorithm uses the same basic concept from the FP tree of getting the combinations of the nodes in the FP tree. It considers three cases in finding out the frequent item sets. It makes comparison between the frequency in the new improved FP tree F and the support S given by the user. The three possible cases are equal to, less than and greater than. Based on the different cases it decides for the frequencies of the frequent item sets and also the nodes to be considered for getting combinations. Let us see each case in detail. We will represent the frequency of the item in new improved FP tree as q.F and the support by q.S. For the first case when the q.F is less than the q.S then the frequency IF will be q.F+C as for this case q.F is less we will consider the count from the node table, because it may happen that an item will have less frequency in the tree but in actual may have higher value of the frequency in the database. The frequent item set will be generated as all the possible combinations of item considered and all intermediate nodes up till the most frequent item node in New Improved FP-tree, because the intermediate nodes in the tree represent the correlation among the items. For the second case

when q.F is equal to q.S the frequency IF will be q.F. Since, in this case it is satisfying the constraint for the minimum support. The frequent item set will be generated as all the possible combinations of the item considered and nodes having higher frequency in New Improved FP-tree. For the third case when q.F is greater than q.S then IF will be q.F. The frequent item set will be generated as all the possible combinations of item and its parent node in New Improved FP-tree, because parent node will always have higher frequencies then the child node.

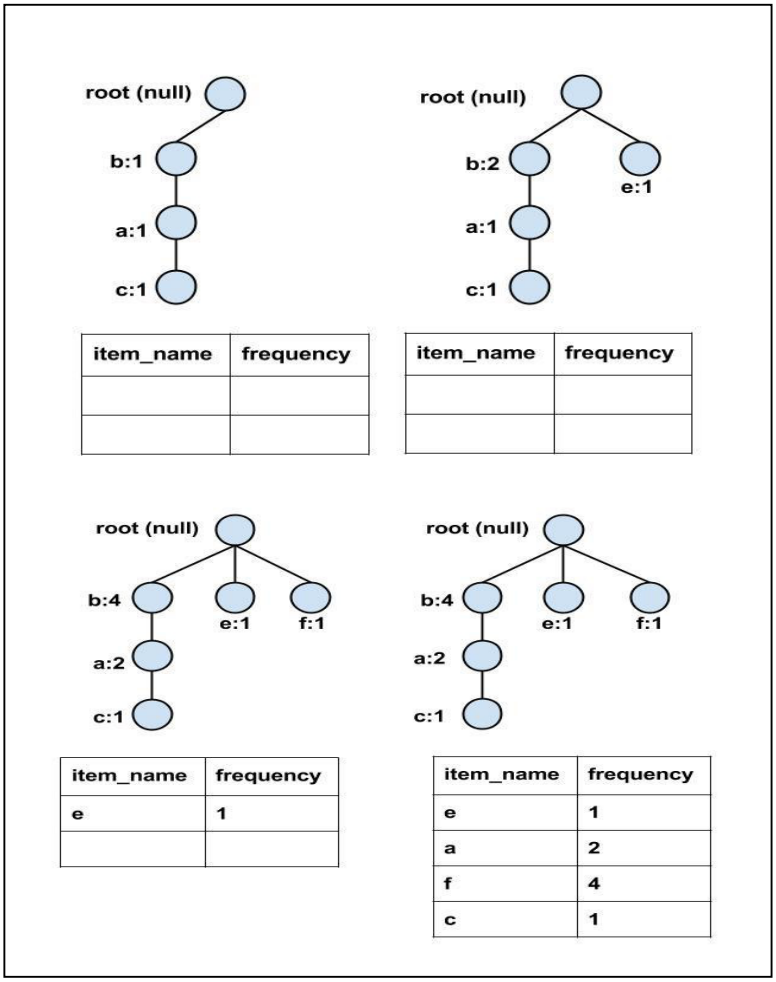

Fig. 2. Construction of the new improved FP tree.

Let us further consider table-1 database and apply the algorithm-1 and algorithm-2 on the same to get the frequent item sets. Fig.2 shows the output for the algorithm-1 i.e. the tree generated and the node table generated. Table-3 shows the output of the algorithm-2 i.e. the frequent item sets generated for each node.

After the application of the algorithm-2 we can apply the concepts given at the section iii to get the association rules. The frequent item set along with its frequency and the user input of support and confidence will finally give the association rules.

Table 3. Frequent Item set.

Item	Frequent item set
b	{b:7}
a	{a:4};{a,b:4}
c	{c:2};{c,a:2};{c,b:2};{c,a,b:2}
f	{f:2};{f,a:2};{f,b:2};{f,b,a:2}
e	{e:2};{e,b:2}

5. Results

For the experiment we consider the data given at table-1. The parameters that we have considered for comparison are database scan, candidate generation, conditional pattern bases, and conditional FP trees. The graphs below show the comparison of Apriori, FP growth, Proposed algorithm with different parameters. The x-axis represents the procedure and y- axis represents the no. of time the procedure has occurred.

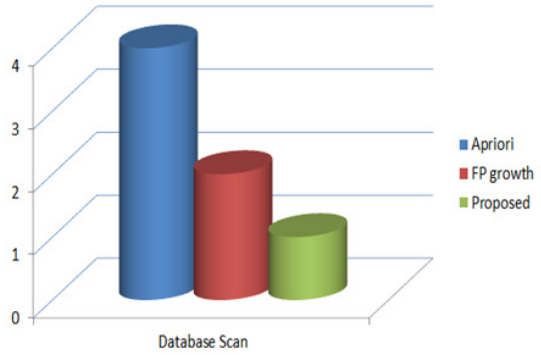

Fig. 3. Comparison graph in terms of database scan.

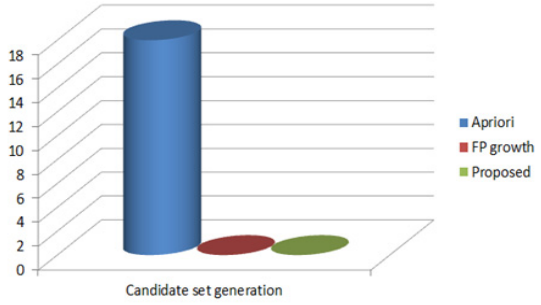

Fig. 4. Comparison graph in terms of Candidate set generation.

The Fig. 3,4,5,6 shows the graphs for the comparison of existing and proposed algorithms. The Fig. 3 shows the comparison in terms of database scan. Apriori scans the database 4 times, FP growth 2 times and proposed algorithm only 1 time. The Fig. 4 shows the comparison in terms of the candidate set generated. Apriori generates the candidate set 18 times, FP growth does not generate any candidate set and proposed algorithm also does not generate any candidate set. The Fig. 5 shows the comparison in terms of the conditional FP trees generated. Apriori generates does not generate the conditional FP trees, FP growth generates it 26 times and our proposed algorithm does not generate any conditional FP trees. The Fig. 6 shows the comparison in terms of the conditional pattern bases generated. Apriori algorithm does not generate any conditional pattern bases, FP growth generates the pattern bases 10 times and the proposed algorithm generates the conditional pattern bases 5 times.

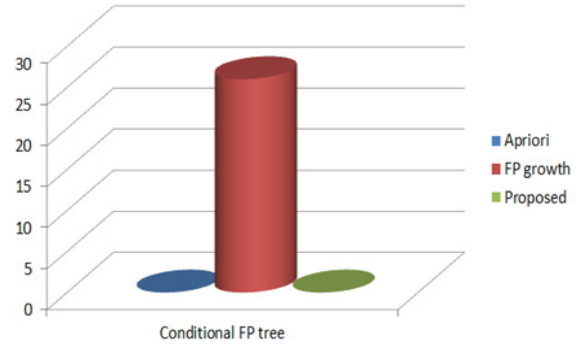

Fig. 5. Comparison graph in terms of Conditional FP tree.

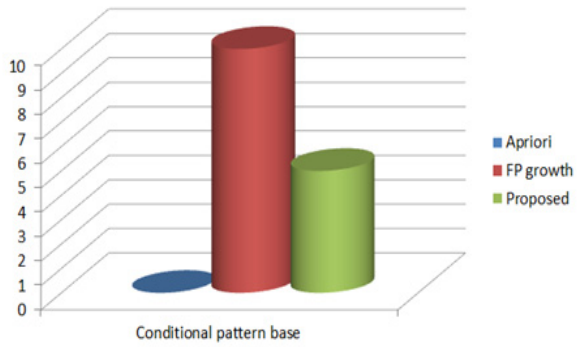

Fig. 6. Comparison graph in terms of Conditional pattern bases.

From the results we can easily analyze that the proposed algorithm is very efficient. It reduces the no. of scans to the database, as a result of which time is saved. It generates no candidate set as compared to Apriori algorithm. It does not generate the conditional FP trees which saves a lot of memory, as in traditional FP growth algorithm a huge no. of conditional FP trees are generated. It also generates less no. of conditional pattern bases because it considers only most frequent item since most of the rules of the interest are associated with the most frequent item in the database. All these points prove the efficiency of the proposed algorithm.

6. Conclusion

Association rule mining has a mentionable amount of practical applications, including classification, XML mining, spatial data analysis, and share market and recommendation systems. It is the task of finding out interesting rules from the databases which would help benefit the business in making profit and would also help in many other areas of its applications by different ways. Many techniques have been developed so far in this area. We have developed a new technique which overcomes the limitation of FP growth algorithm which is the generation of conditional FP trees. Due to this,by using our technique we savea lot of memory which was earlier spent in storing the huge set of conditional FP trees.

In this work, with the help of the a new improved FP tree and a new Frequent Item set mining algorithm, we are able to save a lot of memory in terms of reducing the no. of conditional pattern bases and conditional FP trees generated. We are also able to reduce the no. of times the database is scanned. We conducted the experiments on the test dataset and found that the developed technique is efficient than the existing system. Also the new improved FP tree gives the correlation among the items more clearly.

As the future work on the system, we can work on XML data with this technique. There is a need for a standard for exchange of data over the web and representing semi-structured data. By using XML both can be achieved. The demand for mining XML data on the web is increasing. Although many techniques have been presented over time for XML data mining, however, the barrier for further development is simplicity of the technique and efficiency. As the future work, it would be interesting to work on XML data with the developed technique.

Acknowledgements

All the authors would like to thank head, department of computer engineering (DJSCOE). I would also like to thank research guide Meera Narvekar for her valuable comments and help towards this research work.

References

1. Agrawal R, Imielinski T and Swami A, "Mining association rules between sets of items in large database",Proceeding of the 1993 ACM SIGMOD International Conference on Management of Data, ACM Press, Dec. 1993, pp. 207-216.
2. Yan Hu, Ruixue Han, “An Improved Algorithm for Mining Maximal Frequent Patterns”, 978-0-7695-3615-6/09/2009 IEEE.
3. Li Juan and Ming De-ting, “Research of An Association Rule Mining Algorithm Based on FP tree”,978-1-4244-6585-9/10/2010 IEEE.
4. Jun TAN, Yingyong BU and Bo YANG, “An Efficient Close Frequent Pattern MiningAlgorithm”, 2009 Second International Conference on Intelligent Computation Technology and Automation IEEE.
5. Syed Khairuzzaman Tanbeer, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, Young-Koo Lee, “Efficient single-pass frequent pattern mining using a prefix-tree”, 2008 Elsevier,Information Sciences 179 (2008) pg. 559–583.
6. Ramakrishnan Srikant, Quoc Vu and Rakesh Agrawal, “Mining association rules with item constraints”, American Association forArtificialIntelligence (www.aaai.org), 1997.
7. Qihua Lan,Defu Zhang, Bo Wo, “A new algorithm for frequent item set mining based on Apriori and FP-tree”, Global Congres on Intelligent System, 2009.
8. C. Silverstein, S. Brin, and R.Motwani, “Beyond Market Baskets: Generalizing Association Rules to Dependence Rules”, Data Mining and Knowledge Discovery, 2(1), 1998, pg 39–68.
9. S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, “Dynamic Item set Counting and Implication Rules for Market Basket Data”, SIGMOD '97 Proceedings of the 1997 ACM SIGMOD international conference on Management of data, Pages 255-264.
10. Jiawei Han, jianpei, and Yiwen Yin, “Mining frequent patterns without candidate generation”, Paper ID: 196, SIGMOD ‘2000.
11. Rakesh Agrawal and Ramakrishnan Srikant, “Fast algorithms for mining association rules”, Proceedings of the 20th VLDB Conference Santiago, Chile, 1994.