IJCSC Volume 5 • Number 1 March-Sep 2014 pp. 154-163 ISSN-0973-7391

Frequent Pattern mining Using Novel FP-Growth Approach

Sanjay Patel[1], Dr. K.Kotecha[2], Viral Patel[3], Harsh Shukla[3], Dhyey Shah[3], Sharan Raghu[3]
[1] Vishwakarma Government Engineering College, Chandkheda, Ahmedabad, Gujarat
[2] Institute of Technology, Nirma University, Ahmedabad, Gujarat
[3] CE Dept, Vishwakarma Government Engineering College, Chandkheda, Ahmedabad

Abstract: With the advancement in technology, information is exponentially increasing day-by-day. With such abundant information, methods are required to analyse the welter of information and extract fruitful knowledge. With the help of data mining, information obtained can be used in a number of applications like product evaluations, medicine & science and marketing. Data mining has main three types: Classification, Clustering and Association Rule Mining. Frequent Pattern Mining is a part of Association Rule Mining. Various methods like Apriori, ECLAT, CTU-Mine, CT-ITL, FP-Growth, CATS Tree, Can Tree etc. have been developed to find out the Frequent Patterns. In this paper, extension of FP-Growth approach for Frequent Item-set Mining; a sub section in Association Rule Mining where the most frequent item-set from the database is obtained is explained. At the end of the paper this novel approach is compared with FP-Growth and Apriori algorithm.
Keywords: Association Rule Mining, Data Mining, Frequent Item-Set mining, FP-Growth.

1. INTRODUCTION

Nowadays, the amount of data that is created every day is estimated to be 2 exabytes. It was estimated that in 2007 it was not possible to store all the data that was being produced. This massive data stimulates new challenging discovery tasks. Different ways to automatically analyse, summarize, classify, and characterize trends in these data are required. Data processing is one of the most active and interesting areas of the database research community. There are many areas of research including statistics, visualization, artificial intelligence and machine learning, which contribute to this field.

One of the greatest challenges face today is making sense of all this data. Knowledge discovery is the process of identifying new patterns and insights in data, whether it is for understanding the Human Genome to develop new drugs, for discovering new patterns in recent Census data to warn about hidden trends, or for understanding customers better at an electronic web store in order to provide a personalized one-to-one experience.

A. DATA MINING

Data mining or knowledge discovery is the process of analysing data from different perspectives and summarizing it into useful information [1]. Above line includes words data, knowledge and information which may sound confusing at first glance and seems to be same thing but they are different. Data are any facts, numbers or text that can be processed by a computer. The patterns, associations, or relationships among all this data can provide information and Information can be converted into knowledge about historical patterns and future trends. In short, data mining is a process of extraction of useful patterns from data sources like databases, texts, web, images, etc.

In data mining, Association Rule Learning is a popular and well researched method for discovering interesting relations between variables in large databases. Association rules are usually required to satisfy a user-specified minimum support and a user-specified minimum confidence at the same time. Frequent Item set mining is the first step of Association Rule Mining. Initially, the concept of frequent itemset mining is introduced and how other fields are contributing for building new methods for the same problem. With the explosion of information recently, requirements have changed. Content analysis on dynamic data is in trend. Also, parallel processing is must.

B. THE PROBLEM

Association rule generation is usually split up into two separate steps:

1. Minimum support is applied to find all frequent itemsets in a database.
2. These frequent item sets and the minimum confidence constraint are used to form rules.

While the second step is straightforward, the first step needs more attention. Association rules are usually required to satisfy a user-specified minimum support and a user-specified minimum confidence at the same time [2].

C. DATA MINING APPROACHES

Based on the data available and the identified study, we can determine the best data mining approach. Options to choose from include:

i. Induction or Predicting
ii. Clustering or Segmentation
iii. Associations
iv. Sequence
v. Deviation identification

The Data Mining Process is explained in figure1 [3].

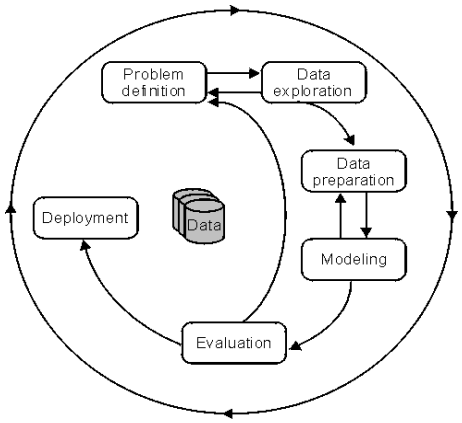


Figure 1[3]

D. GOAL

The main objective is to build an efficient and scalable algorithm for incremental mining problem for frequent item sets. In prior approaches like Apriori algorithm and FP-Growth, whole process is to be re-initialized from base and is not suitable for dynamic analysis. Also, Apriori is redundant as it scans the data set repeatedly for each item set. However, candidate set generation is still costly, especially when there exist prolific patterns and/or long patterns. Here aim is to develop the algorithm using a more efficient approach that uses concept of tree based on FP-growth algorithm. Goal is to minimize the cost of finding frequent item set using this new approach.

2. BACKGROUND

Previous approaches to build frequent itemsets were following brute force techniques as in Apriori and improving that using Frequent Pattern tree. Literature suggests that this problem is NP-hard. The time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows.

Frequent Item-Set Mining Approaches

i) Apriori

The Apriori Algorithm is an influential algorithm for mining frequent itemsets for Boolean association rules [4].

Frequent Itemsets: The sets of item which has maximum support (L_i for i^{th} itemset).

Pseudo-code:

C_k: Candidate itemset of size k
L_k: frequent itemset of size k

```
        L1={frequent items};
for (k=1;Lk!=Null;k++) do begin
        Ck+1 = candidates generated from Lk;
        for each transaction t in database do
                increment the count of all candidates in Ck+1 that are contained in t
                Lk+1 = candidates in Ck+1 with min_support
        end
Return k Lk;
```

Drawback: This principle suffers from redundancy in processing. The whole data set is to be scanned again and again for each set of candidate generations.

Apriori works well except when:

- ✓ Lots of frequent patterns
- ✓ Big set of items
- ✓ Low minimum support threshold
- ✓ Long patterns
- ✓ Repeated database scans costly

ii) FP-Growth

Here, frequent item-sets are discovered without candidate item-set generation. It is a 2 step process.

- The first step builds a data structure called the FP-tree. This tree is built using 2 passes over the data-set.
- The second step involves the extraction of frequent item-sets directly from the FP-tree via traversal.

Advantages:

- Only 2 passes over the data set
- Compresses data-set
- No candidate generation
- Faster than Apriori

Disadvantages:

- FP-tree may not fit in memory.
- The tree is expensive to build.
- Time consuming if support threshold is high

iii) FELINE Algorithm with CATS Tree

Cheung and Zaiane[6] designed the CATS tree mainly for interactive mining. The CATS tree is an extension of the FP-tree which improves the storage compression, and allows frequent-pattern mining without the generation of candidate item sets. A CATS tree must be as compact as possible. The idea of tree construction is as follows. It requires one database scan to build the tree. New transactions are added at the root level. At each level, items of the new transaction are compared with children (or descendant) nodes. If the same items exist in both the new transaction and the children (or descendant) nodes, the transaction is merged with the node at the highest frequency level. The remainder of the transaction is then added to the merged nodes, and this process is repeated recursively until all common items are found. Any remaining items of the transaction are added as a new branch to the last merged node. If the frequency of a node becomes higher than its ancestors, then it has to swap with the ancestors so as to ensure that its frequency is lower than or equal to the frequencies of its ancestors.

iv) Canonical Tree (CanTree)

Leung, Khan and Hoque[7] proposed CanTree solves the problems/ weaknesses of the FELINE or AFPIM algorithms as follows:

(i) Items are arranged according to some canonical order that is unaffected by the item frequency. Hence, searching for common items and mergeable paths during the tree construction is easy. No extra downward traversals are needed during the mining process.

(ii) The construction of the CanTree is independent of the threshold values. Thus, it does not require such user thresholds as preMinsup.

(iii) Since items are consistently ordered in CanTree, any insertions, deletions, and/or modifications of transactions have no effect on the ordering of items in the tree. As a result, swapping of tree nodes is not needed.

3. PROPOSED WORK

Initially, only the number of times a set of items occur in transaction was stored but when a traversal method for solution was developed, a problem was encountered while building an exact solution of item sets involving more than two items. So, a proposition was needed to remember the transaction in which a particular set of items occurred. By remembering the number of transactions with item sets while building model, more information while scanning data once could be stored hence repeated data scans could be avoided for each item sets as in Apriori algorithm.
Next question arose like what type of model to create? Some of the options were to use Tree, Graph or Hyper-graph functions. Using hyper-graph, the graphical data representation was good but the numbers of comparisons were equivalent to Apriori approach due to repeated scanning of data. For tree, the data is scanned only once and dynamic changes can be made easily.

Other issues:

- Constructing a tree directly from data set needed two scans as in FP-growth approach.
- Tree constructed from matrix was not optimal. Tree traversal requires reconstructing the tree in an optimal way.

A. ALGORITHM

For the transaction database
For each row representing a transaction
Until new transactions are available DO
    Prepare a matrix representing each transaction as a column and each item as a row
        If the transaction in column 'j' contains item in row 'i' then
            Set the cell at i,j to '1'
        Else
            Set the cell i,j to '0'

From the Matrix
Make groups of transactions in each column having sequence of 1's and set the count to 1
Make an ArrayList of all these groups
Now combine all the similar groups and merge them and make count of the first such group as the total count.
From these groups make TreeNodes and set various properties like parent, parenttrace, count and node name.
Store this TreeNodes in an ArrayList.

Make a node called "Root"
Till there are elements in the ArrayList
Add the nodes as child of "Root" and so on according to the properties of those nodes.

Traversal:
First start from bottom of the tree
For each node
If there is parent having same initial name as that node
    Count is added to parent node
    And if the count is more than support name of node is saved as answer
Else
    Traversal continues

Second time traversal is from top
In this case each subtree of root is treated separately
For each subtree
Child having different name then "parent" are considered and each itemset in that node including subtreeitemset are checked for further combination with "parent" node and if combination is found then count is incremented and if count is more than support it is saved as answer.

B. EXAMPLE

Consider following example. Each line shows a different transaction where a space is used to separate two items. Consider support: 3.

Table 1: Example of problem Instance

Transaction No.	Items in each transaction
1	B C D F G
2	D F G
3	F G A C
4	B D F G E
5	A G F E
6	D E F G

Step 1:

Scan the data and form a matrix with columns as transaction number and rows as item set id. The matrix has a '1' at row '*i*' and column '*j*' if we have the item set represented by row '*i*' in transaction '*j*'.

Table 2: Matrix Formed from data in table 1

Transactions	1	2	3	4	5	6
B	1			1		
C	1		1			
D	1	1		1		1
F	1	1	1	1	1	1
G	1	1	1	1	1	1
A			1		1	
E				1	1	1

Here, item B is present in transaction 1 and 4, so we place a '1' in respective position in the matrix.

Step 2:

Count the number of '1's in each row and re-arranges the rows in a descending order based on that count. Also discard the rows having the count less than the **support**.

Table 3: Count of itemsets using matrix shown in table 2

Transaction	Count
B	2
C	2
D	4
F	6
G	6
A	2
E	3

As minimum support of 3, after A, B and C are eliminated. This procedure is also known as **Pruning**.

Table 4: Itemsets sorted according to count

Transaction	Count
D	4
F	6
G	6
E	3

Sorted Matrix after **Pruning**:

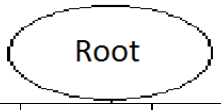
Table 5: Sorted Matrix after Pruning

Transactions	1	2	3	4	5	6
F	1	1	1	1	1	1
G	1	1	1	1	1	1
D	1	1		1		1
E				1	1	1

Step 3:

Now scan row by row and make group of consecutive 1's in column. And also find similar group of 1's that has same parent trace and have same row as ending. We will also construct a tree with the scanning of rows as soon as we find a group (or one or more similar groups). Consider that root node is already been constructed.

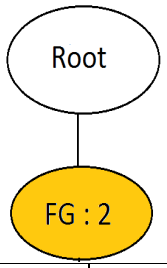
1st row:



Transactions	1	2	3	4	5	6
F	1	1	1	1	1	1
G	1	1	1	1	1	1
D	1	1		1		1
E				1	1	1

Here no group is made.

2nd row:

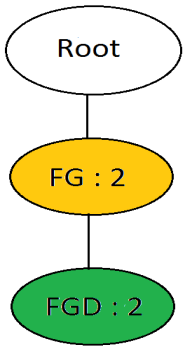


Transactions	1	2	3	4	5	6
F	1	1	1	1	1	1
G	1	1	1	1	1	1
D	1	1		1		1
E				1	1	1

Here as shown two group are constructed which are group of column 3 and 5 are same. As there is no node except root node we will construct node(s) as necessary

according to groups. Here both are similar groups so we will show no of count of similar group and show it in only one node. So the node will be named "FG" as consecutive 1's are of row F and G. and count as "2" because there are two groups.

3rd row:

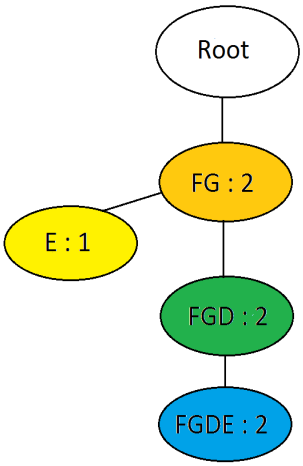


Transactions	1	2	3	4	5	6
F	1	1	1	1	1	1
G	1	1	1	1	1	1
D	1	1		1		1
E				1	1	1

Here new group of column 1 and 2 are formed. So we will show it in tree. Now as there are two groups so as they both are same we will construct one node named "FGD" as the consecutive 1's are of row F, G and D. now to attach this node we will have to first find out if any existing node having same initials i.e. in our case "F" or "FG" then we will set this node as parent node to new node. Here one condition is that their parent trace according to matrix would be same. Means here both node's group are starting from "F" so their parent is "ROOT".

4th row:

Transactions	1	2	3	4	5	6
F	1	1	1	1	1	1
G	1	1	1	1	1	1
D	1	1		1		1
E				1	1	1



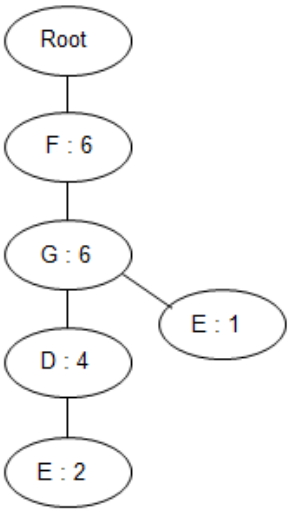


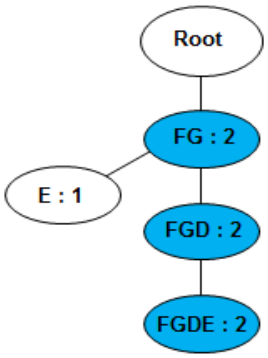
Figure 2: Tree constructed using FP-growth approach (left) and proposed approach (right)

Here 3 new group in column 4, 5 and 6 are made. Now as groups of column 4 and 6 are same so we will construct a node "FGDE" and count as "2" and having parent trace as ROOT. Now we will find from tree if any existing node having parent trace "ROOT" and match the initials of name. We will select node having maximum initial matching node as parent to new node.
Also a group is made at column 5. No similar group are there so we will construct a node having name "E" and count "1". Now we will search in tree if any node having same parent trace. As here no parent trace is matched then we will attach this node to the node having matching name from reverse with reverse parent string(not whole parent trace need to be matched maximum string matching node is selected) and join new node as child to that node.

Traversal:

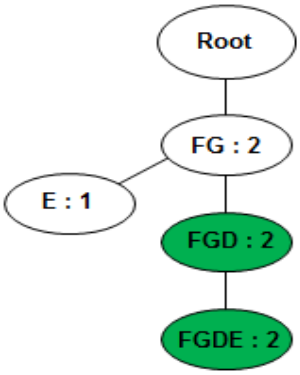
Once the tree has been constructed, we can get the count of the frequent itemsets by traversing the tree in the following manner. We can obtain the count of the frequent itemsets FG by adding the count of all the nodes having FG in their names. So we get FG as
FG:2 + FGD:2 + FGDE:2 => FG:6.

TREE



Similarly, we can find the count of FGD by adding the counts of FGD and FGDE so we get FGD:4.

TREE



Similarly, we get the count of FG, GD, GE, DE, FGD, FGE, and GDE. The table shows the final answer.

Table 6: Final Answer from traversal

Frequent Itemsets	Measured Support
➢ D, F, G	4
➢ D, G	4
➢ D, F	4
➢ F, G, E	3
➢ F, E	3
➢ F, G	6
➢ G, E	3

4. RESULTS

In these experiments, transaction databases generated by IBM [11] are used in computer system having core 2 duo 2.0 GHz processor, 160 GB hard disk and 2 GB RAM. The goal of experiment is to find out the performance of proposed algorithm over existing algorithms. In figure 7, it shows comparison between Apriori, FP-growth and Extension of FP-growth by requiring time for different min_sup value. Results show that Extension of FP-growth requires minimum time as compared to Apriori and FP-growth. The Apriori algorithm works on the principle of candidate generate and test, so it requires the maximum execution time. Figure 3 shows the comparison for the Database with the time and Figure 4 shows the same for minimum support and time.

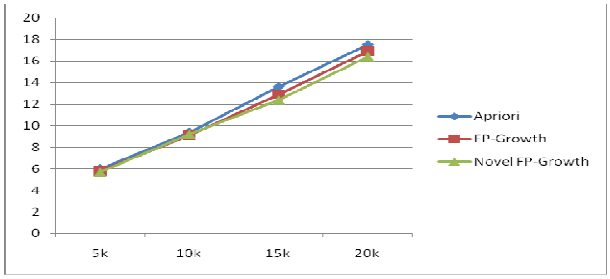


Figure 3: Database Vs. Time

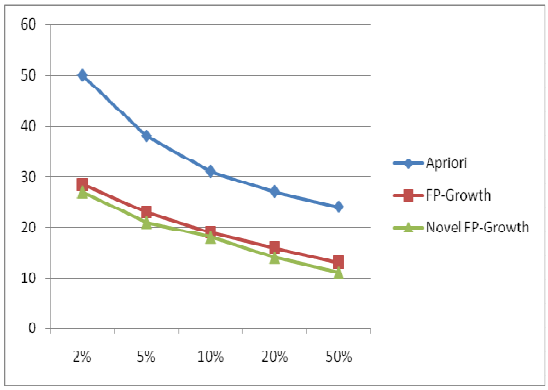


Figure 4 : Min_Sup Vs. Time

5. CONCLUSION

Frequent Item-set Mining problem can be successfully implemented with our new approach based on tree. Initially, implicit pruning is applied so the item-sets with count less than support factor are removed. Then explicit pruning is performed to remove the patterns after traversal. Here, we observation shows that after explicit pruning optimal answers can be obtained, i.e. possible frequent item sets with minimum support. To get the exact answer, which item set are frequently occurring and their count, adopt a traversal algorithm and store it in the answer list.

REFRENCES:

1. B.Palace (1996), Data Mining : "Technology Note prepared for Management 274A " Anderson graduate school of Business management at UCLA. Spring 1996.
2. Q. Zhao et al. (2003), "Association Rule Mining: A Survey "Technical Report, CAIS, Nanyang Technological University, Singapore, No. 2003116 , 2003.
3. Google Images - Data Mining Process.
4. A. Wasilewska (2013), " Apriori Algorithm: Basics" Lecture Notes
5. F. Verhein (2008), " Frequent Pattern Growth Algorithm : An Introduction " ,School of Information Technologies The University of Sydney, Australia
6. W. Cheung and O. R. Zaiane (2003), "Incremental Mining of Frequent Patterns without Candidate Generation of Support Constraint", University of Alberta, Edmonton, Canada.
7. C. Leung et al. (2005), " CanTree: A Tree Structure for Efficient Incremental Mining of Frequent Patterns" Fifth International Conference on Data Mining.
8. S. Tanbeer et al.(2008), " CP-Tree: A Tree Structure for Single-Pass Frequent Pattern Mining" Department of Computer Engineering, Kyung Hee University, Springer-Verlag Berlin Heidelberg 2008, pp 1022-1027.
9. Li Liu et al. (2007)," Optimization of Frequent Itemset Mining on Multiple-Core Processor", VLDB '07, September 23-28, 2007, Vienna, Austria.
10. G Liu et al (2013)," A Flexible Approach to Finding Representative Pattern Sets" , IEEE Transactions on Knowledge and Data Engineering, 2013.
11. C.K.S. Leung(2007)," CanTree: a canonical-order tree for incremental frequent-pattern mining", Knowledge and Information Systems, April 2007, Volume 11, Issue 3, pp 287-311
12. J. Han (2006), " Data Mining: Concepts and Techniques, Second Edition, University of Illinois at Urbana-Champaign.
13. A. Rajaraman et al. (2010), " Mining of Massive Datasets " Stanford University.
14. J. Han et al (2004)," Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach", Kluwer Academic Publishers. Manufactured in The Netherlands. Data Mining and Knowledge Discovery, 8, 53–87, 2004