MINING OF FREQUENT POSITIVE OPINIONS BY USING MATHEMATICAL TECHNIQUES

P. DHANA LAKSHMI, M.LAVANYA
Assistant Professor, CSSE Dept, Assistant Professor (SL), M.C.A Dept, Sree Vidyanikethan Engg.college
A.Rangampet
mallidhana@gmail.com, lavanya479@rediffmail.com

ABSTRACT
In recent years the sizes of databases has increased rapidly. This has led too growing interest in the development of tools capable in the automatic extraction of knowledge from data. The term Data Mining, or Knowledge Discovery in Databases, has been adopted for a field of research dealing with the automatic discovery of implicit information or knowledge within databases. Several efficient algorithms have been proposed for finding frequent itemsets and the association rules are derived from the frequent itemsets, such as the Apriori algorithm. These Apriori-like algorithms suffer from the cost to handle a huge number of candidate sets and scan the database repeatedly. A frequent pattern tree (FP-tree) structure for storing compressed and critical information about frequent patterns is developed for finding the complete set of frequent itemsets. But this approach avoids the costly generation of a large number of candidate sets and repeated database scans, which is regarded as the most efficient strategy for mining frequent itemsets. Finding of infrequent items gives the positive feedback to the Production Manager. In this paper we are finding frequent and infrequent itemsets by taking opinions of different customers by using Dissimilarity Matrix between frequent and infrequent items and also by using Binary Variable technique. We also exclusively use AND Gate Logic function for finding opinions of frequent and infrequent itemsets.

Key words: knowledge discovery, frequent items, infrequent items, similarity, dissimilarity

INTRODUCTION
Finding of associations between items from a large database of business data, has been a latest topic within the area of data mining [1][2]. The effective management of business is significantly dependent on the quality of its decision making. These are useful in market basket analysis and catalog design. It is therefore important to analyze past transaction data to discover customer purchasing behavior and improve the quality of business decision. There are several techniques available to find frequent items. The strategies for mining frequent itemsets, includes Apriori [1], and FPgrowth[3]. To support the above analysis, collect the transaction items based on requirement and store it in a database. The major work of mining frequent itemsets is to find all itemsets that satisfy a certain user-specified minimum support. Each such item set is referred to as large item set. The rest of the paper is organized as follows: Section 2 presents the existing work Section 3 proposes the efficient data mining algorithms for finding opinions of frequent items. Finally, we conclude this paper and present directions for future research in Section 4.

II EXISTING ALGORITHMS IN ASSOCIATION RULES MINING
Association rule mining as introduced in [1] searches for relationship between items in a data set. It finds association, correlation, or causal structures among set of items or objects in transaction databases, relational databases and other information repositories. To mine association rules, database of transaction is needed. And each transaction is list of items. Then apply mining algorithm to find the association rule. Finding frequent itemsets plays an impotent role in the field of data mining. Frequent Itemset is a special for many data mining problems like discovery of association rule correlation [4,5] and sequential pattern [6,7]. As defined in [2], the problem is stated as follows.

Let \( \mathcal{I} = \{i_1, \ldots, i_n\} \) be a set of items. An itemset \( X \) is a subset of items, i.e. \( X \subseteq \mathcal{I} \). A transaction \( T = (\text{tid, } X) \) is a 2-tuple, where \( \text{tid} \) is a transaction id and \( X \) an itemset. A transaction \( T = (\text{tid, } X) \) is said to be contain itemset \( Y \) if and only if \( Y \subseteq X \). A transaction database \( D \) is a set of transactions. The number of transactions in \( D \) containing itemset \( X \) is called the support of \( X \). Given a transaction database \( D \) and a support threshold \( \minsup \), an itemset \( X \) will be called as frequent pattern if and only if \( \sup(X) > \minsup \).

2.1 The Apriori Algorithm
Apriori algorithm finds all frequent itemsets. The first pass of the algorithm simply counts item occurrences to determine the large 1-itemsets. A subsequent pass, say pass \( k \), consists of two phases. First, the large itemsets \( L_{k-1} \) found in the \( (k-1) \)th pass are used to generate the candidate itemsets \( C_k \), using the Aprioricandidate generation function (apriori-gen). Next, the database is scanned and the support of candidates in \( C_k \) is counted. For fast counting, an efficient determination if the candidates in \( C_k \) that are contained in a given transaction \( t \) is needed. A hash-tree data structure [8] is used for this purpose.
2.2 The FP-growth Algorithm
The main bottleneck of the Apriori-like methods is at the candidate set generation and test. This problem was dealt with by introducing a novel, compact data structure, called frequent pattern tree, or FP-tree then based on this structure an FP-tree-based pattern fragment growth method was developed, FP-growth. This approach avoids the costly generation of a large number of candidate sets and repeated database scans, which is regarded as the most efficient strategy for mining frequent itemsets. The definition, according to [9] is as follows.

A frequent pattern tree (FP-Tree) is a tree structure defined below.
1. It consists of one root labeled as “root”, a set of item prefix sub-trees as the children of the root, and a frequent-item header table.
2. Each node in the item prefix sub-tree consists of three fields: item-name, count, and node-link, where item-name registers which item this node represents, count registers the number of transactions represented by the portion of the path reaching this node, and node-link links to the next node in the FP-tree carrying the same item-name, or null if there is none.
3. Each entry in the frequent-item header table consists of two fields, (1) item-name and (2) head of node-link, which points to the first node in the FP-tree carrying the item-name.

III PROPOSED TECHNIQUES
Section II gives the information about only finding frequent items means which are frequently purchased by the customer. Generally in real time applications most of the people collect more information about the products before they are purchasing particular product. The finding of product features using existing mining algorithms is a difficult task. All the existing techniques concentrate on finding frequent or infrequent items only. No such existing algorithms are available to find positive opinions. But so many web sites allow the users to give their opinions about the product while they are purchasing or after using the product. Collecting either positive or negative opinions using existing algorithms is a difficult task. But our proposed algorithm can find the positive opinions of frequent items about the products. Before discussing about proposed technique let us first define some definitions:

Symmetric and Asymmetric binary Variables:
A binary variable is Symmetric if both of its states are equally valuable and carry the same weight otherwise it is asymmetric.

Table 1: Notational descriptions of Relational table
<table>
<thead>
<tr>
<th>Easy</th>
<th>Difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price returns</td>
<td>Reasonable</td>
</tr>
<tr>
<td>Noise</td>
<td>Less</td>
</tr>
<tr>
<td>Vibration</td>
<td>Less</td>
</tr>
<tr>
<td>Durability</td>
<td>Long</td>
</tr>
<tr>
<td>Installation</td>
<td>Easy</td>
</tr>
</tbody>
</table>

Table 2: Description of Relational table of Products using binary attributes
<table>
<thead>
<tr>
<th>Product name</th>
<th>Carry</th>
<th>Price</th>
<th>Noise</th>
<th>Vibration</th>
<th>Durability</th>
<th>Installation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Washing Machine</td>
<td>D</td>
<td>R</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>E</td>
</tr>
<tr>
<td>Inverter</td>
<td>E</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>S</td>
<td>D</td>
</tr>
<tr>
<td>Cell phone</td>
<td>E</td>
<td>R</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>E</td>
</tr>
<tr>
<td>Air Conditioner</td>
<td>E</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>S</td>
<td>D</td>
</tr>
<tr>
<td>Refrigerator</td>
<td>D</td>
<td>R</td>
<td>M</td>
<td>L</td>
<td>M</td>
<td>E</td>
</tr>
</tbody>
</table>

Dissimilarity between binary Variables:
One approach involves computing the dissimilarity between all binary variables involves computing dissimilarity matrix. Dissimilarity between two binary variables can be calculated by using the following formula:

\[ d(i,j) = \frac{r+s}{q+2} \]

where \( q \) is the number of variables that equal to 1 for both objects \( i \) and \( j \), \( r \) is the number of variables is equal to 1 for object \( i \) but that are 0 for object \( j \) and \( s \) is the number of variables is equal to 1 for object \( j \) but that are 0 for object \( i \).
Suppose that a Product item set or product relation list table (Table 2) contains the attributes carry, price, noise, vibration, durability, installation. Where Product name represents object identifier and the attributes specified in Table 2 all are Asymmetric attributes. For Asymmetric attributes we can set two values either 0 or 1. Carry related to implicit attribute weight will give two values.

### 3.1 Proposed Algorithm

#### Algorithm 1: (Using Dissimilarity matrix)

1. Construct relation table of the products and their attributes.
2. Find the dissimilarity between each product and their attributes.
3. Compare Dissimilarity of each product with minimum support set for the product item set.
4. If (Dissimilarity of each product > minimum support) then
   4.1 the given item is infrequent
   Else
   4.2 the given item is frequent
end if

5) Repeat step 4 for all products in the product list or product item set.

#### Algorithm 2: (Using AND Gate Logic)

Consider two variables A and B. A represents Product list and B represents Cluster. Maintain all products which are produced to the customers are in A and B contains all positive attributes related to the products.

Example: Pen drive is easy to carry.
Here carry is a attribute and easy is a positive attribute to the above sentence. For the same sentence Difficult to carry is the negative attribute.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

The Logic is as follows

The algorithm consists of the following steps

1) **case1:**
   If product belongs to Product List and if opinion is in the cluster
   Then
   Find the dissimilarity of the product

2) **case2:**
   Else if the product belongs to product list but opinion is not in the cluster
   Find the support of the product

3) **case3:**
   Else if product does not belongs to product list and and opinion is in cluster
   No need to calculate dissimilarity between variables

2) Apply the table 3 on each product in the product list and store the result of each case in a separate table
3) Find the support of each table
3) The result of case1 is for positive opinions
4) Result of case2 is for negative opinions

Both these two algorithms are new techniques to calculate positive opinions about the products.

The problem in Algorithm 1 is
1) Setting of minimum support, it depends on Domain Expert. If it is high, rare number of associations between items gets. If it is low, more number of frequent items will be occurred.
2) It calculates both frequent and infrequent items. So time taken to calculate frequent items will be high because it scans entire database.

But Algorithm 2 calculates and considers only positive opinions.

### IV. CONCLUSION

The Proposed algorithm for discovering opinions of frequent item sets based on dissimilarity matrix using binary variable is a new method and is found efficient when compared to Apriori and FP-tree. As such, we are still working on it with the aim of extending the application of this algorithm to various kinds of databases.
REFERENCES


[9] J. Han, J. Pei, Y. Yin. “Mining Frequent Patterns without Candidate Generation”. Proc. of ACM-SIGMOD, 2000.