Graph Based Approach for Finding Frequent Itemsets to Discover Association Rules

Manju
Research Scholar, Dept. of Computer Science & Applications, Kurukshetra University, Kurukshetra

Dr. Chander Kant
Assistant Professor, Dept. of Computer Science & Applications, Kurukshetra University, Kurukshetra

Abstract—The discovery of association rules is an important task in data mining and knowledge discovery. Several algorithms have been developed for finding frequent itemsets and mining comprehensive association rules from the databases. The efficiency of these algorithms is a major issue since a long time and has captured the interest of a large community of researchers. This paper presents a new approach that can mine frequent itemset or patterns in less time and in a straightforward way. Majority of the algorithms developed for finding frequent itemsets scan the database repeatedly and are based on the concept of minimum threshold support value. The proposed approach is based on graph and finds frequent itemsets without repeatedly scanning the database. The algorithm finds frequent itemsets irrespective of support level and can be used for finding the largest most frequent itemset. The proposed approach performs single scan of the database in first phase and draws a graph in which edges are labeled with the respective transactions ids. In second phase, a table is constructed which has all distinct labels and the corresponding itemsets. The largest most frequent itemset is selected from this table according to the given selection criterion.

Index Terms—data mining, association rules, knowledge and data management, database

1. INTRODUCTION

The advancements in technology and computation power, in addition to computerization of most of the commercial activities, have given rise to huge amount of data. Availability of powerful database systems and the ease of managing and using such databases, has also contributed to the growth of extensively large data repositories. This in turn demands equally powerful tools and techniques to be used for extracting information and knowledge. As a result, data mining has become an interesting research area [1]. Data mining is the process of mining useful, comprehensive, previously unknown and interesting information from the stored data. By comprehensive we mean that the mined information should be wide-ranging that is hidden in the data and provides some facts and information that can be used further for management decision making and process control [2]. The task of data mining includes methods that mine different types of knowledge from databases. The kind of knowledge that can be exposed includes association rules, classification rules, generalizations and summarizations etc. The discovery of association rules is the area that has fascinated many researchers for finding innovative approaches to detect hidden and interesting associations that exists in the collected data. Several algorithms have been developed for mining association rules that are significant and provide important information of planning and control. The work reported in this paper is a step for finding association rules quickly from databases in a simple and straightforward way.

2. RELATED WORK

Association rule mining is one of the important and well researched techniques of data mining to find important correlations among data items.

The first algorithm to generate all frequent itemsets was proposed by Agrawal et al. [3] and named AIS (after the name of its proposers Agrawal, Imielinski and Swami). The algorithm generates all the possible itemsets at each level of traversal. Thus it generates and stores frequent as well as infrequent itemsets in each pass. Generation of infrequent itemsets was undesirable and was a major drawback over its performance. Later on AIS was improved upon and renamed as Apriori by Agrawal et al. The new algorithm uses a level-wise and breadth-first search for generating association rules. Apriori and Apriori Tid algorithms generate the candidate itemsets by using only the itemsets found large in the previous pass and without using the transactional database. Apriori uses the downward closure property of the itemset support to prune the itemset lattice- the property that all subsets of frequent itemsets must themselves be frequent [4].

A similar algorithm called Dynamic Itemset Counting (DIC) was proposed by Brin et al. DIC partitions a database into several blocks marked by start points and repeatedly scans the database. Unlike Apriori, DIC can add
new candidate itemsets at any start point, instead of just at the beginning of new database scan. At each start point, DIC estimates the support of all itemsets that are currently counted and add new itemsets to the set if all its subsets are estimated to be frequent [9].

Vu et al. proposed a rule based prediction technique to predict the user featured location, but this method generates more candidate itemsets than required. As the database must be scanned multiple times the algorithm was expensive in terms of run time and I/O load [5]. The problem of multiple scanning was improved upon by using a compact tree structure and finding frequent itemsets directly from the data structure. The algorithm scans the database twice. The first scan of the database discards the infrequent itemsets and the second pass constructs the tree. The method encodes a dataset using a compact data structure called FP-tree and extracts the frequent itemsets directly. This FP-growth algorithm proposed by Han et al. gained lots of popularity [6].

To improve upon the cost of main memory requirement Pei et al. proposed another algorithm called H-mine using array and tree based data structure. A distinct feature of this method is that it has very limited and precisely predictable space overhead and runs really fast in memory-based setting. It can be scaled up to very large databases by database partitioning and when the dataset becomes dense, FP-tree can be constructed dynamically as part of the mining process [7].

Sahaphong and Boonjing proposed a new algorithm which constructs a pattern base using a new method that is different from the patterns base in the FP-growth and mined frequent itemsets using a new combination method without the recursive construction of a conditional FP-tree [12].

Another approach based on FP-tree and co-occurrence frequent items (COFI) was proposed by V.K. Srivastava et al. to find frequent itemsets in multilevel concept hierarchy by using a non-recursive mining process [13].

Toivonen proposed the sampling algorithm for finding association rules to reduce database activity. The sampling algorithm applies level-wise method to sample and search for frequent itemsets in sample that can be done in main memory and so only one scan of transactions is required. The algorithm uses a lower minimum support threshold to mine the frequent itemset [8].

The Pincer-Search algorithm was proposed by Lin et al. in 1997 and can efficiently discover the maximum frequent set. The Pincer-Search combines both the bottom-up and top-down directions. The main search direction is still bottom-up but a restricted search is conducted in the top-down direction. This search is used only for maintaining and updating the new data structure designed for this study, namely, the maximum frequent candidate set. It is used to prune the candidates in the bottom-up search. Another very important feature of this algorithm is that it is not necessary to explicitly examine every frequent itemset. Therefore it performs well even when some maximal frequent sets are long. This algorithm reduces the number of times the database is scanned and the number of candidates considered [10].

The parallel algorithms for the discovery of association rules using clustering techniques to approximate the set of potentially maximal frequent itemsets was first proposed by Zaki et al in 1997. This algorithm uses two clustering schemes based on equivalence classes and maximal hypergraph cliques and study two lattice traversal techniques based on bottom-up and hybrid search and also use vertical database layout to cluster related transactions together [11].

Most of the algorithms that mine frequent itemsets use a horizontal data layout. However many researcher use a vertical layout. The Eclat algorithm proposed by Zaki et al. generates all frequent itemsets in a breadth-first search using the joining step from the Apriori property when no candidate items can be found. The Eclat algorithm is very efficient for large data sets but is less efficient for small data sets [14].

The partition algorithm for mining association rules that is different from the classical algorithms. It requires two database scan to mine frequent itemsets and also reduce I/O overhead. First the partition algorithm scans the database once to generate a set of all potential frequent itemsets and in the second scan their global support is obtained. The algorithm partitions the database into small chunks which can be held in main memory. The partitions are considered once at a time and all large itemsets are generated for that partition. These large itemsets are further merged to create a set of all potential large itemsets [15].

3. DATA MINING AND ASSOCIATION RULES

Data mining is the nontrivial process of extracting useful, unknown and hidden information from data in the databases. It is also called knowledge discovery in databases, which means that interesting knowledge, regularities, or high-level information can be extracted from the relevant sets of data in databases that can be investigated from different angles. Large databases, thus, serve as rich and reliable sources for knowledge generation and verification [16]. Different types of methods and techniques are needed to mine different kind of
knowledge. Based on the type of knowledge to be mined, the task of data mining can be divided into several tasks such as summarization, classification, clustering, association and trend analysis. Association rules are used to discover the relationships, and correlations among items or attributes. These rules can be very effective in revealing unknown relationships and hidden patterns, providing results that can be used for forecasting and decision making.

3.1 ASSOCIATION RULE MINING (ARM)

Association rule mining is one of the most important and nontrivial tasks of data mining. It aims to mine interesting relationships, frequent patterns, associations or casual structure between sets of items in a transactional database. These rules have been proven to be very useful tools for an enterprise as it strives to improve the competitiveness and profitability [17]. Mining association rules from the databases has attracted the attention of large community of researchers in the past few years [4], [6]. The formal model or statement of the problem was given by Agrawal et al. in 1993. To define the problem of ARM let I=I1,I2,…..In be set of n distinct items in the database D. Let T be the transaction that contains a set of items from I. The database contains different transaction records. An association rule is an implication in the form of X → Y, where X, Y are sets of items from I and are called itemsets and the sets X and Y are disjoint sets having no common items. X is called the antecedent and the Y is called consequent and rule is called X implies Y.

3.2 INTERESTINGNESS MEASURES

There are two important measures for finding the interestingness of association rules. These are – support and confidence [18]. Support of an association rule is the fraction of transactions that contain both X and Y to the total number of transactions. Confidence is defined as the fraction of transactions that contain X and Y both to the total number of transactions that contains X. Because of the large volume of data in the database the users are interested in only those items that are purchased frequently. Thus a threshold value is usually defined for support and confidence to drop those rules that are not of much interest. Since mining association rules may require scanning the database repeatedly, the amount of processing could be huge and performance improvement becomes an essential concern [16].

4. DISCOVERING FREQUENT ITEMSETS

The most time consuming operation in the process of discovering association rules is the computation of the frequency of interesting subsets of items. The task of mining useful association rules is essentially to find strong rules. This task can be divided into two major phases. The first phase finds all of the itemsets that satisfy the minimum support threshold, which are the frequent itemsets. The second phase is the rule generation, in which high confidence rules are generated from the frequent itemsets found in the previous phase [18]. It is found that the overall performance of mining association rules is determined by the first step. Once the large itemsets are identified, the corresponding association rules can be derived in a straightforward way. Thus efficient computation of large frequent itemsets is the major concern in discovering association rules. Several approaches for discovering the frequent itemsets have been proposed all with their pros and cons [6] [7] [10] [12]. Most of the early approaches have two major bottlenecks. First the database has to be scanned several times which is time-consuming and second enormous large itemsets are generated and lots of CPU time is wasted in calculation of the support repeatedly.

5. PROPOSED ALGORITHM

As discussed earlier, the most time-consuming and important part of the rule discovery process is the discovery of large frequent itemsets. The database is arranged as set of records of transactions and kept on the disk. To find out the frequent itemsets the database is scanned many times calculating the support of different sets of items and comparing with the threshold value of support. To reduce the database scan and to generate frequent itemsets in a straightforward way, a new approach based on graph is proposed to identify the frequently occurring sets. The approach works in two phases: Graph Construction and Generation of itemsets corresponding to the graph. The major steps in the approach are outlined in the following sub-sections.

5.1 GRAPH CONSTRUCTION

As a first step, a graph of the database items is constructed. Items are represented by nodes and the nodes belonging to the same transactions are connected by an edge between them. This edge is labeled by the corresponding transaction Id. If the nodes (items) corresponding to same transactions have already been connected by an edge, then new transaction Id is added to the existing label. Thus the label of an edge contains all the transaction Ids in which the two items (end nodes of the edge) are part of the corresponding transactions. Let there be five items A,B,C,D,E and the database has four transactions T1, T2, T3 and T4 (or 1,2,3 and 4).

Each transaction has some items. The transactions and the corresponding items of the sample database are shown in Table 1.
There are three possible unordered pairs of items (nodes):

1. First transaction $T_1$ contains three items $A$, $C$, and $D$.
2. Second transaction $T_2$ contains three items $B$, $C$, and $E$.
3. Third transaction $T_3$ contains three items $A$, $B$, $C$, and $E$.
4. Fourth transaction $T_4$ contains two items $B$ and $E$.

After the graph is constructed by the above process, a table is created for finding the most frequent itemset. The table has six columns with headings: Label, Length (L), Frequency (F), $L \times F$, Edges, and Itemset as shown in Table 2.

### 5.2 Generation of Itemsets

After the graph is constructed by the above process, a table is created for finding the most frequent itemset. The table has six columns with headings: Label, Length (L), Frequency (F), $L \times F$, Edges, and Itemset as shown in Table 2.

#### Table 1: Sample Database

<table>
<thead>
<tr>
<th>Transaction Id</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>$A, C, D$</td>
</tr>
<tr>
<td>$T_2$</td>
<td>$B, C, E$</td>
</tr>
<tr>
<td>$T_3$</td>
<td>$A, B, C, E$</td>
</tr>
<tr>
<td>$T_4$</td>
<td>$B, E$</td>
</tr>
</tbody>
</table>

The graph is constructed as follows:

i) Draw as much nodes in the graph as the items in the database and represent each node with item name.

ii) Repeat the following steps for each transaction:
   a. Find all possible unordered pairs of nodes corresponding to the items of the same transactions.
   b. Draw an edge between all such pair of nodes.
      - If new edge is drawn, label this edge with the corresponding transaction Id.
      - If for some pair of nodes, the edge is already drawn, expand its label by adding the corresponding transaction Id.

The graph for Table 1 is shown in the Figure 1. First transaction $T_1$ contains three items $A$, $C$ and $D$. Corresponding to this transaction, there are three possible unordered pairs of items (nodes) - $(A, C)$, $(A, D)$, and $(C, D)$. The edges with end nodes $(A, C)$, $(A, D)$, and $(C, D)$ are assigned Label $<1>$ corresponding to transaction id $T_1$.

Second transaction $T_2$ contains three items $B$, $C$ and $E$. The unordered pairs of nodes corresponding to this transaction are $(B, C)$, $(B, E)$, and $(C, E)$. The three edges in the graph with end nodes $(B, C)$, $(B, E)$, and $(C, E)$ are assigned Label $<2>$. Similarly for transaction $T_3$, six edges are drawn in the graph with end nodes $(A, B)$, $(A, C)$, $(A, E)$, $(B, C)$, $(B, E)$, and $(C, E)$. The edges $(A, B)$ and $(A, E)$ are drawn for first time, so these edges are labeled with Label $<3>$. The edges $(A, C)$, $(B, C)$, $(B, E)$, and $(C, E)$ have already drawn while processing transactions $T_2$ and $T_3$, so the labels for these edges are updated and the corresponding transaction id 3 is appended to the existing labels, resulting in the new labels for these edges $<1, 3>$, $<2, 3>$, $<2, 3>$ and $<2, 3>$ respectively. For transaction $T_4$, the corresponding edge $(B, E)$ is also already drawn, and hence the transaction id 4 is appended to the existing label $<2, 3>$ updating it to $<2, 3, 4>$. 

#### 5.2 Generation of Itemsets

After the graph is constructed by the above process, a table is created for finding the most frequent itemset. The table has six columns with headings: Label, Length (L), Frequency (F), $L \times F$, Edges, and Itemset as shown in Table 2.
For the above table, the attributes are defined as below:

i) Label: It represents the label of the edges generated by the algorithm proposed in Section 5.1.
ii) Length (L): It denotes the length of the Label. It is the number of the transaction IDs which are part of the Label.
iii) Frequency (F): It is the no. of occurrences of the Label in the graph.
iv) L×F: It represents the selection criterion for finding the frequently occurring item sets.
v) Edges: This column represents the edges that have been labeled with the given Label.
vi) Itemset: It represents the corresponding itemset. It contains distinct items which are part of the corresponding Edges.

5.3 FINDING THE FREQUENT ITEMSETS

Method for finding frequent itemsets is straight forward. Once the table is constructed, frequent itemsets can be found by analyzing the proposed selection criterion of L × F. The row having highest value of L × F gives the largest and the most frequent itemsets. In our example, the highest value of L × F is given by the second row of the Table 2, thus the most frequent largest itemset is \{B, C, E\}. This frequent itemset is computed irrespective of any given threshold. The same itemset is computed in [16] when Apriori algorithm, with support threshold value of 2, is applied to the sample data as given in Table 1. If one is interested to find the most frequent itemset of length (size) 2, it can be easily selected from the Table 2. The most frequent itemset of length 2 is \{B, E\} represented by first row of the table. It corresponds to the highest value of L × F for the itemsets having length 2.

6. EFFICIENCY OF THE PROPOSED APPROACH

The proposed approach is efficient from the point of view of database scans. Most of the previous methods perform repeated scans of the database which, in turn, waste CPU time and results in degradation in the overall performance. Another criterion for measuring the efficiency is the effect of change of support threshold value. Several previous methods including Apriori requires the complete procedure to be repeated again if the user specified threshold value is changed. But the present approach is free from such drawbacks. Once the graph and the table for finding frequent itemsets are constructed successfully, we can find the most frequent itemsets of any given length (for threshold value changed) without any sort of rework. The proposed approach when applied to the sample databases given in [16], [19], [20], reported the same frequent itemsets in single database scan, as produced by their respective approaches in multi database scans.

Using the proposed approach, even the next most frequent itemsets can be found which are not discussed in the previous approaches. The next most frequent itemset corresponds to next highest value of L × F for any given length. The proposed approach can be applied directly on small and medium size databases. For large databases, vertical partitioning can be used as a preprocessing step to filter out the infrequent items. This preprocessing step will make the numbers of nodes and the edges manageable.

7. CONCLUSION

Finding the frequent itemsets is a major phase of association rule mining. This paper presents a new approach for finding the frequent itemsets. The approach is based on graph. The proposed approach overcomes two major drawbacks of previous approaches. The method adopted to identify largest frequent itemsets is simple and straightforward. A graph is used to represent the entire database. The graph can be constructed by scanning the database only once. Changing the minimum support (threshold value) does not affect the constructed graph or the table and thus no additional work is needed. Also unlike other approaches, the approach can find frequent itemsets without generating the candidate itemsets and thus reduces the number of repeated database scans reducing the run time and memory consumption. The method is also good where the database is updated frequently, because adding new transactions or items just require addition of few nodes and edges in the graph and amendments in the table.

REFERENCES


