Volume: 4 Issue: I Month of publication: January 2016
DOI:

www.ijraset.com Volume 4 Issue I, January 2016
IC Value: 13.98 ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

A New Approach for Finding Maximally Frequent Set in Transactional Database

Rahul [1], Sharad Chauhan[2], Kamal Sharma[3]
[1]*M.Tech Student,* [2]*Assistant Professor,* [3]*Professor Department of CSE*
E-Max Group of Institutions, Ambala, Kurukshetra University, India1

Abstract-Data mining is a methodology with the ability to extract information from large data sets and transforming it into understandable form for further use. The information obtained is of great value and has proven to be advantageous in various business applications. Apriori algorithm is one of the most fascinating and thoroughly investigated area in the field of data mining. It is used to identify frequent item sets in a transactional database. There exist many implementations for this algorithm using different data structures and methods for generating candidate sets. In our work instead of generating candidate sets and scanning the entire transactional database multiple times, we will be introducing the concept of base table and we will be scanning the entire transactional database only once. Beginning with item set containing one element and gradually increasing size of item set, we have created base tables. These tables contain subsets of fixed size. Later, we apply Binary search on array of base tables. In order to reduce the large size of transaction database, we use map (STL) to record frequency of all distinct transaction. For the purpose evaluating frequent item set, we use longest common subsequence algorithm. It improves the performance of proposed algorithm to a great extent over classical Apriori algorithm. Our algorithm gives better performance than most of the versions of Apriori algorithm proposed till date for transactional database which have dense matrix representation.

Index Terms -- Data Mining, Transaction Database, A-Priori Algorithm

I. INTRODUCTION

Data mining is an interdisciplinary subfield of computer science. It is the computational process of discovering patterns in large data sets ("big data") involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. we are in an age often referred to as the information age. In this information age, because we believe that information leads to power and success, and thanks to sophisticated technologies such as computers, satellites, etc., we have been collecting tremendous amounts of information. Initially, with the advent of computers and means for mass digital storage, we started collecting and storing all sorts of data, counting on the power of computers to help sort through this amalgam of information. Unfortunately, these massive collections of data stored on disparate structures very rapidly became overwhelming. This initial chaos has led to the creation of structured databases and database management systems (DBMS)[1]. The efficient database management systems have been very important assets for management of a large corpus of data and especially for effective and efficient retrieval of particular information from a large collection whenever needed. The proliferation of database management systems has also contributed to recent massive gathering of all sorts of information. Today, we have far more information than we can handle: from business transactions and scientific data, to satellite pictures, text reports and military intelligence. Information retrieval is simply not enough anymore for decision-making. Confronted with huge collections of data, we have now created new needs to help us make better managerial choices. These needs are automatic summarization of data, extraction of the “essence” of information stored, and the discovery of patterns in raw data.

A. Data Mining and Knowledge Discovery

With the enormous amount of data stored in files, databases, and other repositories, it is increasingly important, if not necessary, to develop powerful means for analysis and perhaps interpretation of such data and for the extraction of interesting knowledge that could help in decision making. Data Mining, also popularly known as Knowledge Discovery in Databases (KDD), refers to the nontrivial extraction of implicit, previously unknown and potentially useful information from data in databases. While data mining and knowledge discovery in databases (or KDD) are frequently treated as synonyms, data mining is actually part of the knowledge discovery process[2].

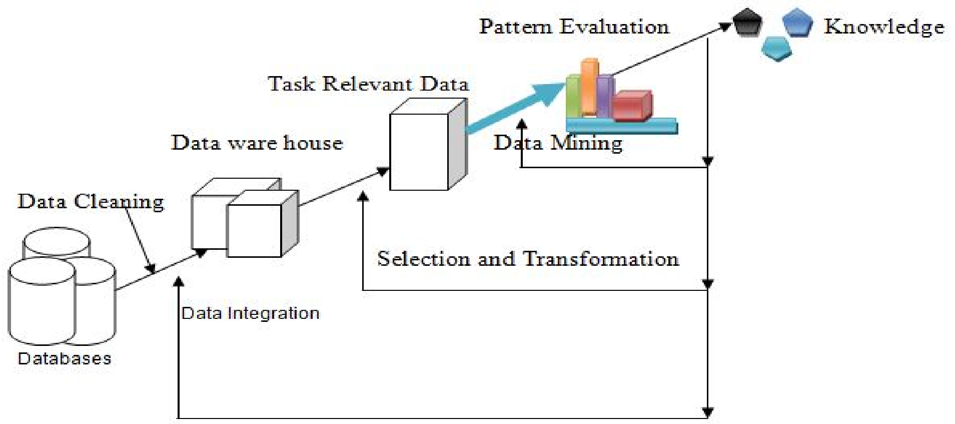

Figure 1.1: The Process of Knowledge Data Discovery

The Knowledge Discovery in Databases process comprises of a few steps leading from raw data collections to some form of new knowledge. As we know information technology is growing and databases generated by the companies or organizations like telecommunications, banking, marketing, transportation, manufacturing etc are becoming huge. It is important to explore the databases and efficiently and completely as data mining helps to identify information in large amount of databases. KDD is the process designed to generate data that shows the well-defined relationship between the variables. It is the process designed to generate data that show the well-defined relationship between the variables. KDD has been very interesting topic for the researchers as it leads to automatic discovery of useful patterns from the database. This is also called as Knowledge Discovery from the large amount of database. Many techniques have been developed in data mining amongst which primarily Association rule mining is very important which results in association rules. These rules are applied on market based, banking based etc. for decision making.

II. VARIOUS WELL KNOWN METHODS FOR ASSOCIATION RULE MINING

A. Apriori Algorithm

Apriori algorithm [1], [9] is one of the classical algorithm proposed by R.srikant and R.agrawal in 1994 for finding frequent patterns for bollean association rules. Apriori employs iterative approach known as level-wise search, where k item sets are used to explore k+1 –item sets. First, set of frequent 1-itemset L1 is found. Next, L1 is used to find frequent 2-itemset L2. Then L2 is used to find frequent 3-itemset 3. Apriori algorithm finds frequent item sets from candidate item sets. It is executed in two steps. Firstly it retrieves all the frequent item sets from the database by considering those item sets whose support is not smaller the the minimum support (min_sup). Secondly it generates the association rules satisfying the minimum confidence(min_conf) from the frequent item sets generated in first step. The first step consist of join and pruning action. While joining the candidate set Ck is produced by joining Lk-1 with itself and pruning the candidate set by applying the apriori property that is all the non empty subset of frequent item set must also be frequent.

B. FP- Growth Algorithm

FP growth algorithm [1], [11], [12] proposed by Jiawei Han finds the association rules more efficiently than apriori algorithm without the generation of candidate item sets. Apriori algorithm requires n+1 scans where n is the length of the longest pattern. FP-growth works on divide and conquer strategy and it requires only two scans of database to find frequent patterns. First, it constructs a FP-tree[12] using the data in transactional database and then mines all the frequent patterns from FP tree. After mining of frequent patterns the association rules can be generated very easily. The pseudo code for FP-growth algorithm is as follows[1],[10]. To sum up, the efficiency of FP tree algorithm account for three reasons. First the FP tree is a compressed representation of the original database because only those frequent item are used to construct the tree, other irrelevant information are pruned. Secondly this algorithm only scans the database twice. Thirdly fp tree uses divide and conquer method that considerably reduced the size of subsequent conditional FP Tree. Every algorithm has its limitations, for FP-Tree it is difficult to be used in an interactive mining system. During the interactive mining process the users may change the threshold of support according to the rules. However for fp tree the changing of support may lead to the repetition of the whole mining process. Another limitation of FP-Tree is that it is not suitable for incremental mining. Since as time goes on databases keep changing, new dataset may be inserted into the database.

Those insertions may also lead to repetition of whole process if we employ FP-Tree algorithm.

C. Partitioning Method

Partitioning method [1] provide the improvement over classical Apriori algorithm. It works in two steps. In first step, it divides the transactions of the database D into n non-overlapping partitions and then finds the support count for each partition. In second step, global frequent item set among the candidates is found. The partitioning method requires only two database scan as compare to n+1 scans required by the Apriori algorithm.

D. Transaction Reduction Method

Transaction reduction method [1] employee a property that a transaction does not contains any k-frequent item sets cannot contain (k+1)- item set. Therefore, such transaction can be removed from the database for further consideration.

E. Hashing Method

Hashing is the method to improve the efficiency of Apriori algorithm. In hashing technique[1] the frequent item sets are found by mapping the frequent items into hash buckets of hashing table. Hashing technique can reduce the size of k-item set Ck. For example, when scanning each transaction in a database to generate the frequent 1-itemset C1, we can generate all the 2-itemset for each transaction, map them into different buckets of a hash table structure and increase the corresponding bucket count. An item set whose bucket value cannot be frequent and can be removed from the candidate set.

III. PROPOSED METHOD

A. Creation of Base Tables

Base tables are created based on number of items in item set. If item set consists of n items then we construct an array of size n and every element in this array contains a table. Every table contains all possible subsequence of specified length. Every element of table consists of two fields, string and frequency count. Initial frequency of all elements of table is set to zero. Let us suppose our item set is: a,b,c,d then we can construct base in following steps:-

Initialize an array of size n to hold tables. Standard template library map is best suited for this purpose as every element of array has to hold a table having two fields, string and frequency count. In map if we use string to integer then default value is set to one. So we need to change it to zero.

Second step is generation of all possible subsequences for given item set. It is required as for creation of base tables all possible subsequences are indispensable. Subsequences are assigned to that element of array where array index is equal to subsequence length.

Now, we have to assign subsequence generated to table. Length of subsequence is used in order to decide subsequence is allocated to which table. For example if subsequence length is one then subsequence is allocated to table contained by array index one. If our item set contains two elements a,b then "a" and "b" is allocated to table contained by array index one. While "ab" is allocated to table contained by array index two. For item set having elements {a,b,c,d} our base table is shown in fig below:-

Table 1.1: Base table

a	0	ab	0	abc	0	abcd	0
b	0	ac	0	abd	0		
c	0	ad	0	acd	0		
d	0	bc	0	bcd	0		
-	-	bd	0				
-	-	cd	0				

As we can see subsequence length and array index which contains table are always equal. This is always followed while allocating subsequence to tables

B. Building Transaction Table

Transaction table is built based on transactional database. We have to scan entire transactional database to build transaction table. In this step we have to use hashing to reduce size of transactional database from billion to where n is number of items in item set.

Building transaction table can be described as follows:-

i) We have to use a textual database for building transaction table. In this textual database we need to have items sorted lexicographically. Sorted lexicographically means if a transaction contains items “cea” then while processing it should be modified to “ace”. Element with precedence alphabetically should come first.

ii) In order to reduce the size of huge transactional database we have to use hashing. We propose use of standard template library map to perform this task. While scanning transactional database if a transaction record does not exist in map we assign its frequency to one else we increment its frequency by one.

iii) Entire transactional database is scanned once in order to build transaction table having frequency count of distinct transactions. It reduces size of transactional database from millions to at most 2n(where n is the number of items).

C. Longest Common Subsequence

Before moving further we need to get acquainted with longest common subsequence algorithm which falls in dynamic programming paradigm. It is used when we have to check whether any element of base table is subsequence of element in transaction table. We need to evaluate length of longest common subsequence and if it is equal to length of element from base table it returns true otherwise false. It is used extensively in areas of genetics.

D. Binary search

Binary search is the most vital part of our proposed algorithm. Ideally binary search is used for searching elements in list or array. In binary search we continue searching process while lower limit is less than equal to upper limit. In every iteration we evaluate a value mid which is half the sum of lower and upper limit in that iteration.

IV. PROPOSED ALGORITHM

Input: transactions_record- Transactional Record, support- Minimum Support Count & K-Number of items

Output: List of maximal frequent item sets

```
1. String s=domain·substr(0,K)
2. generateSubset (s)
2. buildTransactionTable(transactions_record)
3. ind=binary_search(1,K,sup)
4. If ind==0 then
5.        return null
6. end
7. Map<string,int>::iterator it , i
8. vector<string> v
9. For each (it=m[ind]·begin()) to (it!=m[ind]·end() && b==false) do
10.       it->second=0
11.       For each i=p·begin() to i!=p·end() do
12.                 If lcs(i->first,it->first)==true
13.                           it->second+=i->second
14.           end
15.       end
16. end
17. For each it=m[ind].begin() to (it!=m[ind].end()) do
18. If  it->second>= support then
19.                 v.push_back(it->first)
20.       end
21.  end
22.  return  v
```

V. COMPLEXITY ANALYSIS

Let m be the number of transactions and n be the number of items. In general, m ranges from billions to trillions and we know that,

Volume 4 Issue I, January 2016
ISSN: 2321-9653

time complexity of Apriori algorithm is as:

$T1 = O(m \_ 2n)$ ......(1)

According to our proposed algorithm we can evaluate its time complexity by analyzing its each part separately and summing them up. At first we need to build transaction table by scanning entire transaction database. For this, we can use standard template library like map to filter unique transaction and their frequency. The one prerequisite for this methodology is that transactions should be in lexicographic order. Its time complexity can be given as:

$Tt = O(m \_ log(2n)) = O(m \_ n)$ ......(2)

After building transaction table we need to generate all possible subset in order to build base table. Its complexity can be given as below:

Bt =O (2n) ......(3)

Later we have suggested to apply binary search on these base tables which takes log n time. For every value of mid (obtained in binary search), we have to scan entire base table array containing item set of size equal to mid and apply LCS for every element of base table with every element in transaction table. In worst case scenario when there is no such item set which is frequent then our time complexity can be given as:

$T = O(logn \_ [[C(n; n/2) + C(n; n/4) + \dots C(n; 1)] \_ 2n \_ n2])$ ...(4)

Here C gives binomial co-efficient and it provides number of elements in base table. Now
for every element in base apply LCS on it with respect to every element in transaction table. Transaction table contains at most 2n item set and complexity of applying LCS can be at most n2. Hence worst case time complexity of our proposed algorithm, using (2),(3),(4)can be given as:

$T2 = O(m \_ n + 2n + logn \_ [[C(n; n/2) + C(n; n/4) + \dots C(n; 1)] \_ 2n \_ n2])$

for large values of n, we can say that

$T2 = O(logn \_ [[C(n; n/2) + C(n; n/4) + \dots :: C(n; 1)] \_ 2n \_ n2])$ ......(5)

Evaluating T2/T1 we get,

$T2/T1 = (logn \_ [[C(n; n/2) + C(n; n/4) + \dots :: C(n; 1)] \_ n2])/m$ ......(6)

For considerable values of n (i.e. the number of items) and we know that m ranges from billions to trillions, taking that into consideration equation (6) will always be having a value much smaller than 1. Hence, our proposed algorithm will always give better performance with respect to time variant, as long as the value of m is greater than 2n

VI. RESULTS

A. Overview

We have performed experimental evaluation of proposed algorithm and we have compared it with classical A-Priori algorithm. Proposed algorithm gives better results than classical A-Priori algorithm with respect to time variant.

B. Performance Evaluation

For our analysis, we have used transactional database consisting of two thousand transactions and item set consists of 11 items. The results obtained from the comparison are depicted in the figure given below:

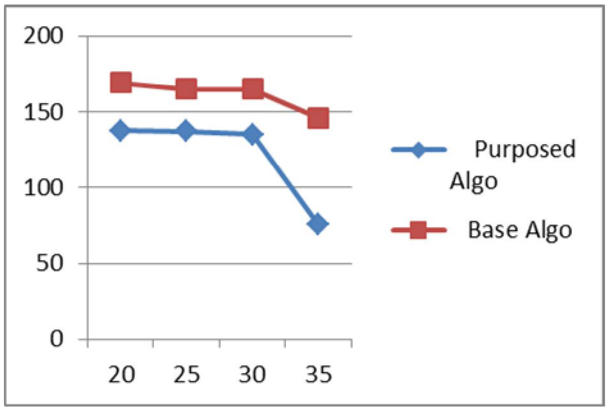

Figure 1.2: Time (Y-axis) v/s Support (X-axis) for dataset 1

For our analysis, we have used transactional database consisting of four thousand transactions and item set consists of 14 items. The results obtained from the comparison are depicted in the figure given below:

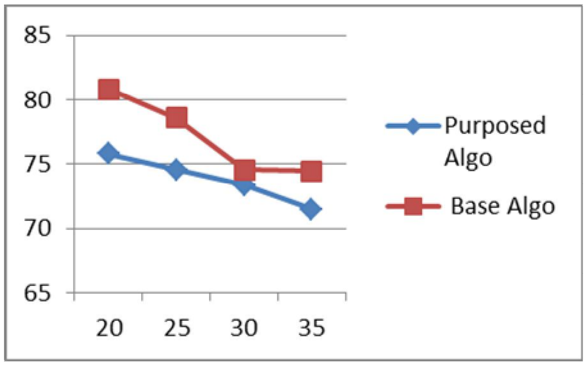

Figure 1.3: Time (Y-axis) v/s Support (X-axis) for dataset 2

VII. CONCLUSION AND FUTURE SCOPE

A. Conclusion

Our proposed algorithm performs much better in comparison to the classical Apriori algorithm with respect to time variant. We need to have lexicographically sorted transactional database to reduce its size using hashing. The size of transactional database gets reduced to 2n (where n is number of items) from billions of transactional records. Instead of scanning level wise iteratively the way we scan in Apriori algorithm we propose a different approach. We propose to use binary search in which we initiate scanning reduced transactional database for subsequences of half the size of item set, if we find any frequent item in that base table we move further reducing overhead of scanning for subsequences of smaller lengths. Therefore, it gives better performance for Considerable number of items, but on the contrary, our proposed algorithm requires more space as compared to classical Apriori. For the purpose evaluating frequent item set, we use longest common subsequence algorithm. It improves the performance of proposed algorithm to a great extent over classical Apriori algorithm. Our algorithm gives better performance than most of the versions of Apriori algorithm proposed till date for transactional database which have dense matrix representation. There exists a tradeoff between the time and space complexity and our algorithm is well suited for time variant. The worst case scenario, i.e., when no item/item-set is frequent, is very rare to occur. In general cases, our proposed algorithm has a far better performance as when we get the frequent item in a base table, binary search is resumed and we move to next base table as assigned by binary search.

B. Future Scope

We can further improve the performance of our proposed work by scanning the transaction table only for those item-sets present in the base table, which are superset of previously identified frequent item-sets but in that case, we have to keep track of all frequent item-sets found while scanning the previous base table, thereby decreasing the time complexity further. In addition to it we can also improve on time variant by using hash table for subset matching as it has linear time complexity instead of longest common subsequence which takes more time.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami, “Mining Association Rules between Sets of Items in Large Database,” Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Vol.22, Issue 2, 1993, pp. 207-211
[2] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules,” Proceedings of the 20th International Conference on Very Large Data Bases, 1994, pp. 487-499.
[3] Chaudhuri S., Dayal U.: An Overview of Data Warehousing and OLAP Technology, SIGMOD Record 26(1): 65-74 ,1997
[4] Han J., Chee S., Chiang J. Y. : Issues for On-Line Analytical Mining of Data Warehouses. Proc. Of 1998, SIGMOD'96 Workshop on Research Issues on DMKD'98 , 1998, pp. 2:1-2:5
[5] Anderberg, M.R. “Cluster analysis for applications”, Academic Press, New York, 1973, pp. 112-113
[6] F. Bodon, “A Fast Apriori Implementation,” In B. Goethals and M. J. Zaki, editors, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Vol. 90 of CEUR Workshop Proceedings, 2003
[7] F. Bodon, “Surprising Results of Trie-based FIM Algorithm,” In B. Goethals, M. J. Zaki, and R. Bayardo, editors, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Vol. 90 of CEUR Workshop Proceedings, 2004
[8] J. Holt and S. M. Chung, "Multipass Algorithms for Mining Association Rules in Text Databases," Knowledge and Information Systems, Vol. 3, No.2, Springer- Verlag, 2001, pp. 110-103.

[9] J. Holt and S. M. Chung, "Mining Association Rules Using Inverted Hashing and Pruning," Information Processing Letters, Vol. 83, No.4, Elsevier Science, 2002, pp. 211-220.

[10] Sotiris Kotsiantis, Dimitris Kanellopoulos ,” Association Rules Mining: A Recent Overview “GESTS International Transactions on Computer Science and Engineering, Vol.32 (1), 2006, pp. 71-82

[11] Qiankun Zhao and Sourav S. Bhowmick,” Association Rule Mining: A Survey”,Technical Report, CAIS, Nanyang Technological University, Singapore, No. 2003111 , 2003

[12] Divya Bansal*1 Lekha Bhambhu ,”Execution of APRIORI Algorithm of Data Mining Directed Towards Tumultuous Crimes Concerning Women” ² ¹M.Tech Scholar ² Assistant Professor Department of Computer Science Department of Computer Science. J.C.D college of Engineering and Technology J.C.D College of Engineering and Technology G.J.U. University of Science & Technology, India G.J.U. University of Science & Technology, India.