An Enhancement Approach for Finding Maximally Frequent set in Transactional Database

Rahul[1], Kamal Kumar Sharma[2], Sharad Chouhan[3]

[1]Student, M. Tech, E-Max group of Institutions, Ambala
[2]Professor, Dept. of ECE, E-Max group of Institutions, Ambala
[3]Assistant Professor, Deptt. of CS, E-Max group of Institutions, Ambala

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 my work instead of generating candidate sets and scanning the entire transactional database multiple times, I am introducing the concept of base table and scanning the entire transactional database only once. This paper describes various well known methods for Association rule mining also describes approaches for finding maximally frequent set

Keywords: Data mining, Database.

I. INTRODUCTION

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 choice 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.

II. VARIOUS WELL KNOWN METHODS FOR ASSOCIATION RULE MINING AND APPROACH FOR FINDING MAXIMALLY FREQUENT SET

Apriori Algorithm

Apriori algorithm finds frequent item sets from candidate item sets. It is executed in two steps.
Firstly it retrieves all the frequent itemsets from the database by considering those item sets whose support is not smaller thea 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 pseudo code for generation of frequent item sets is given below :

```
Ck : Candidate itemset of size k
 Lk : Frequent itemset of size k
{
L1= frequent 1- itemset
For(k=1; k!=NULL; k++)
{
Ck+1 = Join Lk with Lk to generate Ck+1
Lk+1 =Candidate in Ck+1 with support greater than or equal to min support;
}
End;
Return Lk;
```

Generation of frequent itemsets by applying Apriori algorithm

FP- Growth Algorithm

FP growth algorithm [1],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 conquers strategy and it requires only two scans of database to find frequent patterns. First, it constructs a FP-tree[11] 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.

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.

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.

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. PROBLEM FORMULATION

Frequent Item sets

Mining frequent item set from a large database generates a large number of item sets satisfying both minimum support and minimum confidence. To overcome this problem concept of closed frequent item set and maximal frequent item set is introduced.

Maximal Frequent Item set

An item set is called maximal frequent[1] if it is frequent as well as closed i.e. the item set has no superset as frequent. For Example: Let items :{a,b,c,d,e}; Frequent item set :{a,b,c} ; {a,b,c,d}, {a,b,c,e}, {a,b,c,d,e} these are not frequent item sets. Maximal frequent item sets :{a,b,c}

Closed Frequent Item set

An item set is closed if none of its immediate supersets has the same support as the item set [1]. For e.g., if {bread, butter} is an item set that has support = 4, and all of its supersets has support <4, then {bread, butter} is a closed item set. Let {bread, butter, sugar} is superset of {bread, butter} and has support = 4, then {bread, butter} is no more a closed item set.

IV. PROBLEM METHODOLOGY

Proposed algorithm gives better results than Apriori with respect to time variant. Pseudo code of proposed algorithm is given below:

Proposed(transactions_record, min_support, no_of_elements)

```
{
String s=domain_substr(0,no_of_elements);
Gen_subset(s);
Build_transaction_table(transactions_record);
int ind=b_search(1,n,sup);
}
 map<string,int>::iterator it,i; string str;
bool b=false;
for(it=m[ind]_begin();it!=m[ind]_end()                &&
b==false;it++)
{
for(i=p_begin();i!=p_end();i++)
{
if(lcs(i->first,it->first)==true)
{
str=it->first;
b=true;
 }
 }
 }
cout<<str<<endl;
}
Gen_subset(string s)
{
int n=s_length();
if(n==0)return;
int p=pow(2,n);
vector<char> v[p];//Array list
for(int i=0;i<p;i++)
{
for(int j=0;j<n;j++)
{
if(i && 1<<j)
{
v[i]_push_back(s[j]);//Add i narray list }
}
v[i]_push_back('0'); }
} string str[p-1];
int j;
for(int i=0;i<p-1;i++)
{
str[i]=&&v[i+1][0];
```

```
}
for(int i=0;i<p-1;i++)
{
int len=str[i]_length();
m[len][str[i]]=1;
}
}
```

V. RESULT

I have performed experimental evaluation and comparison of proposed algorithm with the classical Apriori algorithm. For my analysis, I have used transactional database consisting of 1 million transactions and item set consists of 15 items. The results obtained from the comparison are depicted in the figure given below:

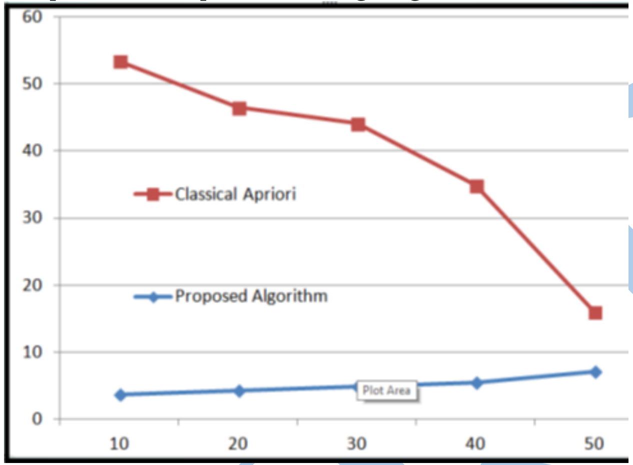

Figure 1.1: Time (Y-axis) v/s Support (X-axis)

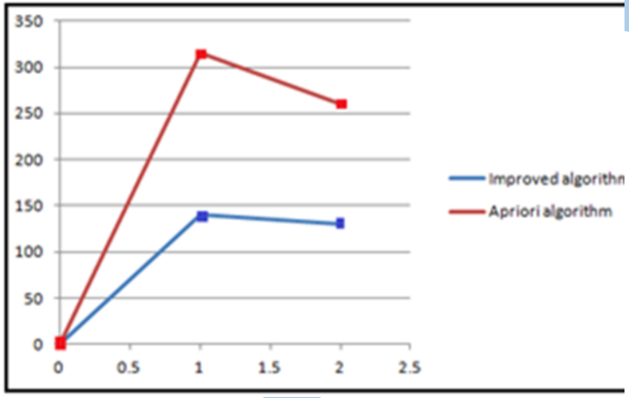

Figure 1.2: (Experimental result of assumed example with different minimum support)

VI. CONCLUSION AND FUTURE WORK

Proposed algorithm performs much better in comparison to the classical algorithm with respect to time variant. I 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. I can further improve the performance of 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

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. 207-216.

[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]. 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

[4]. 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

[5]. 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. 168-183.

[6]. 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.

[7]. J. S. Park, M. S. Chen, and P. S. Yu, "Using a Hash-Based Method with Transaction Trimming for Mining Association Rules," IEEE 1rans. on Knowledge and Data Engineering, Vol. 9, No.5, Sep/Oct 1997, pp.813-825.