

DESIGN AND IMPLEMENTATION OF ALGORITHMS FOR FAST DISCOVERY OF ASSOCIATION RULES

Ashalatha¹, Sinchan², Sowmya Srinivas Majalikal³ and Mrs. Vanitha T⁴

Abstract-Finding a proper frequent itemset is a challenge in database mining. In This paper we are making a survey on already existing algorithm which will give the frequent itemsetlist in a minimum passing throughout the disk and with less time. By using association rule we can find the user behavior. By analyzing the limitation of existing algorithms like partition, DHP and DIC we used a apriori algorithm to find a frequent itemset which takes filtered data to the next iteration. Association rules are commonly used in sales transaction. Association rule will be defined as the relationship between co-occurring items. Experimental results show improvements of finding frequent itemset by taking the limitation of other existing algorithms.

I. INTRODUCTION

Discovering an association rule is the main process in datamining. In large database of sales transactions in mining for association rules between items has been recognized as an important area of database research[5,7]. Finding the association rule can be done by using two steps. The first step is finding all frequent itemsets, i.e., itemsets which occur in the database with a certain user-specified frequency, called minimum support. The second step is forming the rules among the frequent itemsets. Compare to the first step this step is quite simple. In this paper it presents efficient methods to discover these frequent itemsets[1,7,11]. Analyzing the frequent itemset depending on the user request is helpful in main industries. For example in supermarket the user can find the items which are frequently sold in a market. Summary of the database can be calculated by conducting the data mining process. In this paper we are analyzing the frequent itemset using following properties every subset of a frequent itemset is also frequent and itemset are frequent only if that is frequent in one partition.

II. EXISTING ALGORITHMS

A. Direct Hashing and Pruning (DHP) algorithm:

For the next pruning DHP[1,5] collects approximate support of candidates in the previous pass, however, this optimization may be detrimental to performance. All these algorithms make multiple passes over the database, once for each iteration k. To filter a unwanted itemsets for the

¹ Aloysius Institute of Management and Information Technology (AIMIT) St Aloysius College(Autonomous) Mangalore, Karnataka, India

² Aloysius Institute of Management and Information Technology (AIMIT) St Aloysius College(Autonomous) Mangalore, Karnataka, India

³ Aloysius Institute of Management and Information Technology (AIMIT) St Aloysius College(Autonomous) Mangalore, Karnataka, India

⁴ Aloysius Institute of Management and Information Technology (AIMIT) St Aloysius College(Autonomous) Mangalore, Karnataka, India

next item generation. Accumulate information about (K+1) –itemsets in advance way so that all possible (K+1) itemsets of each transaction after some pruning are placed into hash table.

Algorithm:

Table1: The Algorithm for DHP Algorithm.

```

Min=a minimum support;
Set all the bucket of B2 to zero
for all transaction T ∈ D do begin
    insert and count the 1-occurrence of the item in hash tree;
    for all 2-subset Z of T do
        B2[b2(Z)]++;
    end
    L1={c|c.count ≥ S,C is in the leaf node of the tree};
    K=2;
    DK=D;
    While({Z|HBK[Z] ≥ S} ≥ LARGE){
        Candidate_gen(LK-1,BK,CK);
        Set all buckets of BK+1 to zero;
        DK+1=∅;
        forall transaction T ∈ DK do begin
            support_count(T,CK,K,T);
            if(|T| > K )then
                DK+1=DK+1 U {T};
        End
    End
    LK={c ∈ CK | c.count ≥ S};
    K++;
}
Candidate_gen(LK-1,BK,CK);
While(|CK| > 0){
    DK+1= ∅;
    forall transaction T ∈ DK do begin
        support_count(T,CK,K,T);
        if(|T| > K )then
            DK+1=DK+1 U {t}
        End
        LK={c ∈ CK | c.count ≥ S};
        if(|DK+1|=0) then
            break;
        CK+1=Candidate_gen(LK);
        K++;
    }
}

```

}

DHP Algorithm Disadvantages:

- i. To find a frequent itemset we have to perform multiple iterations in the database.
- ii. It requires more memory allocation.

B.Partition Algorithm:

By scanning database twice we can reduce the input and output. It partitions the database into small pieces which can be handled in memory. In the first pass it generates the set of all potentially frequent and in the second pass their common support is generated. I/O can be minimized by considering the small sample amount of data from the database. An analysis of the effectiveness of sampling for association mining was presented in database, by using all the required rules of mining.

Algorithm will execute in two phase,first phase the data is divided into a number of non overlapping partition. At the last of first phase large itemset are merged to generate a set of all potential large itemset. In the second phase the partitioned itemset are divided according to the size which can be accommodated in the main memory.

Partitioning Algorithm Disadvantages:

- i.It is suitable for some amount of data in the database.
- ii.It partitions the data randomly from the dataset.
- iii.While merging the data there may be a loss of frequent itemset.

C.DIC algorithm:

DIC Algorithm is an alternative to AprioriItemset Generation. Itemset added and deleted from dataset in each completion of transaction. Examine the subset which is frequent in all the iteration through the database.It uses only general-purpose DBMS systems and relational algebra operations have been studied , but it is not suitable for specialized purpose.

DIC Algorithm Disadvantages:

- i.It may be expensive.
- ii.To check the proper frequent itemset in the database we have to perform multiple iterations.
- iii.Efficiency of the data will be decreased if the data is in the non-homogeneous form.
- iv.New itemset counting may come late.

Algorithm:

Table2: The Algorithm for DIC Algorithm.

1. Itemset will be marked. Mark all the 1-itemsets with square. Unwanted itemset are ignored.
2. While any square itemsets remain:
 1. Reach the end of the transaction list. Increment the counter value in each transaction if any frequent itemset is found and places in square.
 2. If the minimum support is less than square count place that into dashed square. Subset of the superset will be placed in a square board as a new count.
 3. Once a itemset has been counted from the dash stop counting the further transaction.

D.AprioriAlgorithm:

The frequent itemset which is present in subset also considered as a frequent itemset. Apriori algorithm uses downward closure property. Only the itemsets found to be frequent in the previous iteration are used to generate a new candidate set, P_k . Before inserting an itemset into P_k , Apriori tests whether all its $(k - 1)$ -subsets are frequent. we can eliminate a lot of unnecessary candidates in pruning step.

Table2: The Algorithm for DHP Algorithm.

```

L1={ frequent 1-itemset s };
For (k=2; Lk-1 !=0; k++)
    Pk=Set of New Candidates;
    For all transaction t ∈ D
        For all k-subsets s of t
            If (s ∈ Pk) s.count++;
Lk= {c ∈ Pk |c.count>= minimum support };
Set of all frequent itemsets = Uk Lk;
    
```

AprioriAlgorithm Advantages:

- i.Reducing the number of candidate itemset.
- ii.Generating the itemset efficiently by avoiding repeated itemset and infrequent itemset.
- iii.Reducing number of caparison which will store the candidate in hash structure.

III.EXPERIMENTAL RESULT

To find a frequent itemset we are using Apriori Algorithm:

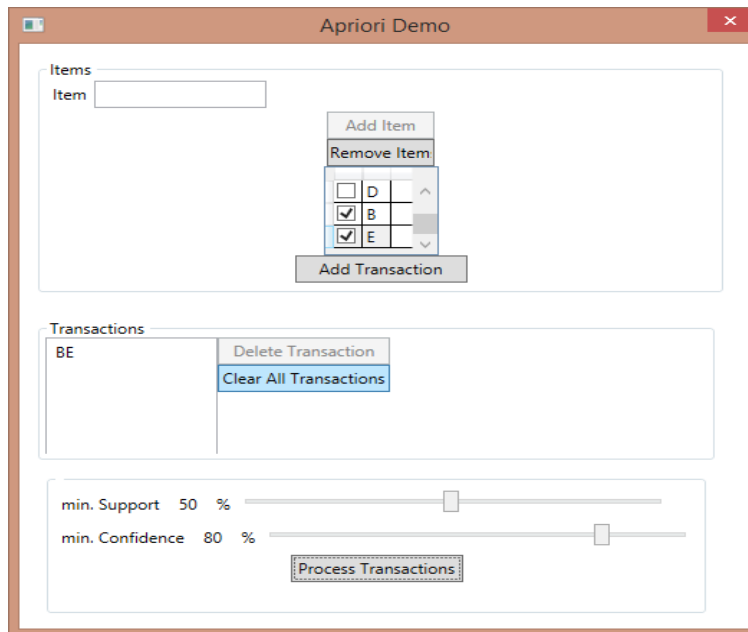
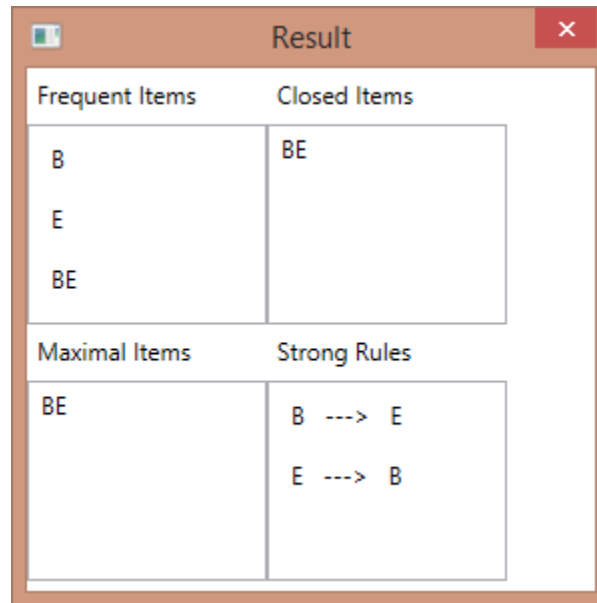


Figure1: Entering the data into the dataset which is used for the transaction.



Frequent Items	Closed Items
B	BE
E	
BE	
Maximal Items	Strong Rules
BE	B ---> E
	E ---> B

Figure 2: Final output consist of frequent itemset

IV.CONCLUSION

By looking into the disadvantages of existing algorithms to find the frequent itemset from the database which consist of huge amount of data. As we conclude in this paper by finding the frequent itemset by using Apriori algorithm which gives the frequent itemset fast compare to other existing algorithm like partitioning, DHC and DCI. The frequent itemset which is present in subset also considered as a frequent itemset. The resultant of the First phase will be passes into the next phase which will decrease the iteration through the database which consist of huge amount of data. So By survey and implementation we conclude that we can find the frequent itemset by using Apriori algorithm Faster than other existing algorithms.

REFERENCES

- [1] J. Han and M.Kambe, "Data Mining: Concepts and Techniques", Morgan Kaufmann Publishers.
- [2] J. D. Holt and S. M. Chung, "Efficient Mining of Association Rules in Text Databases"
- [3] B. Mobasher N. Jain, E.H. Han, and J. Srivastava, "Web Mining: Pattern Discovery from Word Wide Web Transactions" Department of Computer Science, University of Minnesota.
- [4]. C. Ordonez, and E. Omiecinski, "Discovering Association Rules Based on Image Content" IEEE advances in Digital Liabreries.
- [5]. R. Agarwal, T. Imielinski, and A. Swami. "Mining association rules between sets of items in large database".
- [6]. R. Agarwal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Venkamo. "Fast discovery of association rules. In Advances in Knowledge Discovery and Data Mining".
- [7]. R. Bayardo and R. Agrawal. "Mining the most interesting rules".
- [8]. J. Hipp, U. Guntzer, and U. Grimmer. "Integrating association rule mining algorithms with relational database systems".