Review on FI Mining Algorithms

Ms. Bhumika Patel[1],
[1]Department of Computer Engineering
GIDC Degree Engineering College
Gujarat-396406 India

Mrs. Hetal Patel[2],
[2]Department of Computer Engineering
GIDC Degree Engineering College
Gujarat-396406 India

Mrs. Pankti Naik[3],
[3]Department of Computer Engineering
GIDC Degree Engineering College
Gujarat-396406 India

Mr. Viral Patel[4],
[4]Department of Computer Engineering
GIDC Degree Engineering College
Gujarat-396406 India

***Abstract*— Frequent pattern mining is the extraction of interested collection of items from dataset. Frequent itemset(FI) is used for achieving the collection of items according to user's requirement. The researchers have proposed various algorithms like Apriori, Eclat, RElim, SaM etc. In this paper, we are presenting depth analysis of mining algorithms and discuss some problems associated with these algorithms in transactional databases based on vertical, horizontal and hybrid layout.**

Keywords*— *Data Mining, Frequent Item Set Mining, Relim, Support

I. INTRODUCTION

The use of data mining is increasing day by day rapidly and becoming popular in the business environment. Data mining is considered as extracting the knowledge from the large database. The core idea of data mining is to gain useful and unknown information or the patterns from the data in the large dataset. Market basket is a popular method of data mining which aims at finding regularity in the behavior of products purchased by the customers. Frequent item set mining works on the same base i.e. find the item sets that are found frequently as well as together in the transaction set.

Mining of frequent item set is an essential step in association rule induction. Many algorithms like Apriori[1,2], FP-Growth[3], Eclat[4], RElim[5], SaM[6], etc. have been proposed after Agrawal first introducing the problem of deriving categorical association rule from transactional databases[2]. Candidate generate-and-test and pattern growth are two main approaches. Mining algorithms can be categorized based on layout of database: vertical layout, horizontal layout, hybrid layout, etc. Rest of the paper represents brief descriptions of frequent item set mining algorithms and the problem associated with them.

II. FREQUENT ITEMSET MINING

The problem of mining frequent item set was introduced by R. Agrawal, et al[1]. Let I= {i1,i2,i3….,in} be the set of items. A transaction is T = (Tid,I), Where Tid is the transaction identifier. A set of transactions over I is the database D.

A transaction T = (Tid,I) is said to support item set X, called a pattern, if $X \in T \in I$. The no. of transactions in D that containX is called the support of X. X is said to be frequent item set if its support is greater or equal to user defined minimum support. More number of frequent item sets is generated if minimum threshold tends to lower and vice-versa. Thus pruning the infrequent items is the tedious job in mining frequent pattern. Consequently, the main aim of FI mining falls on improving execution-time.

III. FREQUENT ITEMSET MINING ALGORITHMS

This section presents different frequent item set mining algorithms in brief like, Apriori, FP-tree, RElim and SaM.

A. Apriori Algorithm

R.Agrawal was the first who had proposed this Apriori algorithm and it became very popular algorithm for association rule mining. The use of support for removing infrequent candidate item sets is guided by Apriori principles: If an item set is frequent, then all of its subsets must also be frequent and if an item set is infrequent, then all of its supersets must also be infrequent [2].

The database D and user defined minimum support smin is given. This algorithm initially scans the database and support of each item is identified. After completion of this step, the set of all frequent 1-itemsets, will be known and all the infrequent items whose support is less than smin are removed. Thenafter the algorithm iteratively generates new candidate k-itemsets using the frequent (k-1)-itemsets found in the previous iteration.

Table 1 indicates the notations used in Apriori algorithm

TABLE I. NOTATIONS

k-itemset	An item set having k items
Lk	Set of large k itemsets (those with minimum support). Each member of this set has two fields: i) Itemset and ii) support count
Ck	Set of candidate k itemsets (potentially large itemsets). Each member of this set has two fields: i) Itemset and ii) support count
$\bar{C}$ k	Set of candidate k itemsets when the TIDs of the generating transactions are kept associated with the candidates

There are basically two steps: join and prune steps.

- Join: Candidate set is generated by self joining Lk-1 with itself and it is denoted as Ck in join step and this step will generate new candidate k-item sets.
- Prune: Ck is the superset of Lk. Thus it is not necessary that all the members of Ck can be frequent, they may be or may not be the frequent but all k-1 frequent items are included in Ck. In this step, uninterested candidates are removed and all the candidates having their support greater than or equal to smin are considered and it would results in Lk. These two steps are repeated and algorithm terminates when there are no new frequent item sets can be generated.

Apriori algorithms results in great reduction in the size of candidate set but it requires many database scan and leads to performance overhead in case of long patterns and large no. of frequent patterns.

B. *FP-Growth Algorithm*

FP-Growth algorithm is one of the most popular algorithms for finding frequent item sets. It mines frequent item sets without candidate generation, this approach scans the database only twice [3]. This algorithm aims at removing the problem of Apriori algorithm [1,2]. The data structure used for this algorithm is denoted by FP-tree (Frequent-Pattern tree)[7]. FP-Growth algorithm combines in its FP-tree structure a vertical representation and horizontal representation. This algorithm works as follows:

- First scan of the database determines 1-itemsets and their support from horizontal layout and arranges them in support decreasing order as L and infrequent items are removed.
- It constructs the FP-tree containing the root “null”. For each transaction do the following.
  1. Select and sort the frequent items in each transaction of the order of L. Let the sorted frequent itemset in transaction be[p|P], where p is a first item and P is the remaining list.
  2. Insert_tree([p|P],T) function is then called. If T has a child N such that N.item-name = p.item-name, then increment N’s count by 1; else create a new node N and N’s count = 1, N’s parent link be linked to T, and N’s node-link be linked to the nodes with the same item-name via the node-link structure. If P is not empty, then insert-tree (P, N) is called recursively.

FP-Growth algorithm considers only two scans of database. In first scan it collects set of frequent items and in the second scan constructing FP-tree.

C. *SaM Algorithm*

SaM(Split and Merge) algorithm uses purely horizontal representation[6]. This algorithm uses array or link list as a data structure i.e. it is easy to implement and very convenient to execute on external storage if the database to be mined cannot be loaded into main memory. Figure 1 shows basic steps that are followed by SaM algorithm. SaM algorithm works as follows:

- Transaction database is given in origin al form.
- Frequency of each item is identified and infrequent items are removed. Transaction are sorted in ascending order

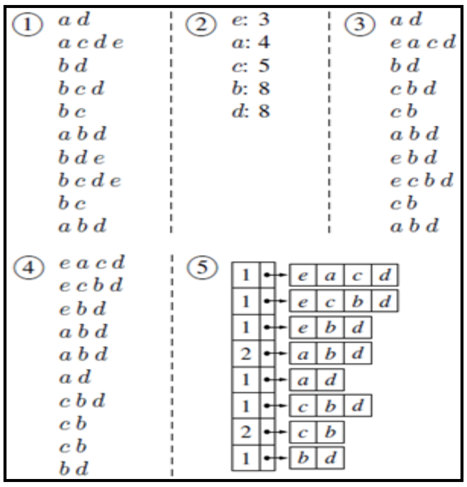

Fig.1. The example database: original form (1), item frequencies (2), transactions with sorted items (3), lexicographically sorted transactions (4), and the used data structure (5)

- Items in each transaction are sorted in ascending order
- All the transactions are sorted lexicographically into descending order.
- The SaM data structure uses two fields: an occurrence counter and a pointer to the sorted transaction.

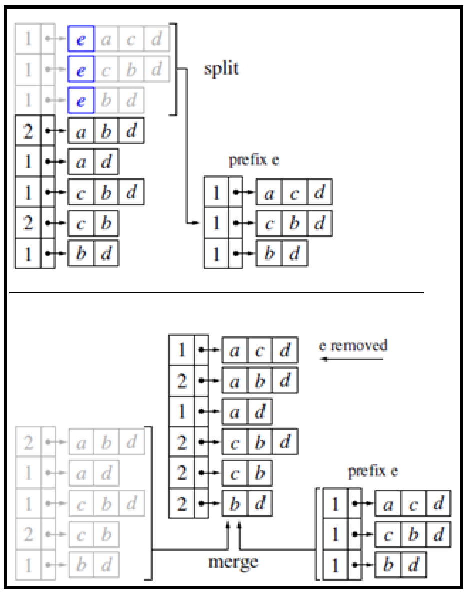

Fig.2. The basic operations of the Split and Merge algorithm: split (above) and merge (below).

The basic operations used by SaM are: split and merge operation. In the split step, an array is split w.r.t. the prefix of the first transaction. In the merge step, this newly created array and rest of original database is merged in the sense that equal transactions are combined and their occurrence counter are incremented by one.

D. RElim Algorithm

RElim(Recursive Elimination) algorithm is inspired by H-mine algorithm and FP-Growth algorithm[3].

This algorithm uses horizontal layout but it groups the transactions w.r.t. their prefix, which is partially, a vertical representation. This algorithm uses array or link list as a data structure.

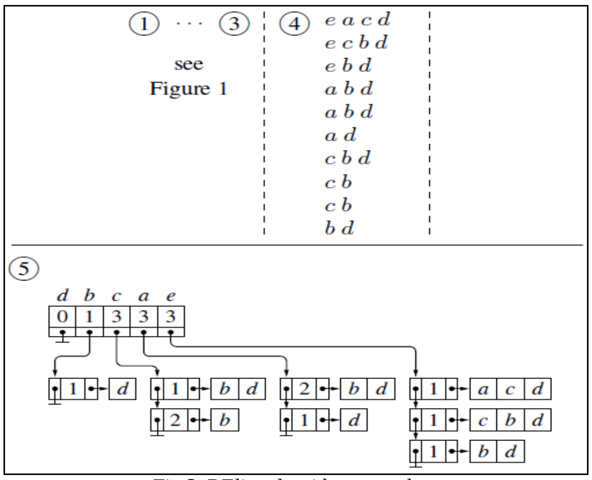

Fig.3. RElim algorithm procedure

Let us understand this algorithm using an example of SaM described above. First three steps are the same as SaM[6]. The major difference is in the presentation of data structure. In the recursive procedure, all the transaction starts with prefix e are considered. As the prefix e contains least support count, thus it is used as an eliminating item to find frequent item sets. After removing prefix e, items of this list are transferred to the list to the right (grey list items) and copies are inserted into the other list(list on right side). This second list is recursively processed to mine frequent item sets.

IV. COMPARISING ALGORITHMS

This section provides the comparison of the algorithms described in the previous section as follows:

- Frequents items are generated using the term minimum support in all the described algorithms.
- Apriori algorithm greatly reduces the size of candidate set but it scans database many times and thus performance is affected. This method is quite successful in market basket analysis.
- FP-Growth overcomes the limitation of Apriori by constructing FP-tree and requires scan of database only twice. As compared to Apriori, the performance of this algorithm is much more improved. But it suffers from memory requirement problem.
- SaM algorithm and RElim algorithms use array or link lisr as a data structure that is easy to implement and free from candidate generation. SaM comes with the advantage that it is well suited for external storage implementation work.

RElim algorithm is the traditional version of SaM, as SaM uses the concept of RElim, RElim works better for sparse datasets, while SaM performs extremely well on dense dataset.

V. CONCLUSION

In this paper, the depth study of the mining algorithms is done and identified many strength and weakness of each. New variants of existing algorithms are compared with classical mining algorithms and results in significant benefits and limitations. This comparison may also fall into various optimization issues that will lead to better performance. Efficiency of the mining algorithms is no longer hindrance but yet there is a need to develop methods to get excellent results.

REFERENCES

[1] R. Agrawal and R. Srikant. "Fast algorithms for mining association rules" In VLDBY94, pp. 487-499.

[2] R. Agrawal, T. Imielinski, and A. Swami. "Mining association rules between sets of items in large databases" In Proc.1993 ACM-SIGMOD Int. Conf. Management of Data, Washington, D.C., May 1993, pp 207-216

[3] J. Han, H. Pei, And Y. Yin. "Mining Frequent Patterns Without Candidate Generation". In: Proc. Conf. On The Management Of Data (Sigmod'00, Dallas, Tx). Acm Press, New York, Ny, Usa 2000.

[4] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New algorithms for fast discovery of association rules" Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD'97, Newport Beach, CA), 283–296. AAAI Press, Menlo Park, CA, USA 1997

[5] C. Borgelt, "Keeping things simple: finding frequent item sets by recursive elimination" Proc. Workshop Open Software for Data Mining (OSDM'05 at KDD'05, Chicago, IL), 66–70. ACM Press, New York, NY, USA 2005

[6] C. Borgelt, "Simple algorithms for frequent item set mining", Springer-Verlag, Berlin, Germany 2010

[7] J. Han, J. Pei, Y. Yin, And R. Mao. "Mining Frequent Patterns Without Candidate Generation: A Frequent-Pattern Tree Approach". Data Mining And Knowledge Discovery, 2003.