Comparative Study of Various Improved Versions of Apriori Algorithm

Shruti Aggarwal[#1], Ranveer Kaur[*2]

[#1]*Dept of CSE, Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India*

[*2]*Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India*

***Abstract*— In Data Mining Research, Frequent Item set Mining has been considered an important task. These item sets leads to the generation of Association rules. These rules tell about the presence of one item with respect to the presence of another item in large dataset. There are efficient methods for generating Association Rules from large databases. This paper describes methods for frequent item set mining and various improvements in the classical algorithm "Apriori" for frequent item set generation.**

***Keywords*— Frequent Item sets, Apriori Algorithm, AIS Algorithm, Association Rule Mining (ARM), Partition Algorithm.**

I. Introduction

There are many databases and data warehouses available all around the world and the major task is to utilize the information or knowledge from these databases. Implicit knowledge within the databases can provide important patterns like association rules which may lead to decision support making, medical diagnosis and many other applications. Association rule mining is task of finding interesting association or correlation relationships among large databases [2].

Association Rules are considered to be interesting as well as useful if they satisfy both a *minimum support* threshold and a *minimum confidence* threshold [1][2][7]. Association Rules can be defined by formal definition as Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items. Let D, be a set of database transactions where each transaction T is a set of items such that $T \subseteq I$. Each transaction must have an identifier, called TID. Let A be a set of items. A transaction T is said to contain A if and only if $A \subseteq T$. An association rule is of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$, and $A \cap B = \varnothing$.

The rule $A \Rightarrow B$ holds in the transaction set D with:

- ***Support; s,*** where s is the percentage of transactions in D that contain $A \cup B$ (i.e., both A and B). This is taken to be the probability, $P(A \cup B)$.
- ***Confidence; c*** in the transaction set D *if* c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability, $P(B|A)$. That is,

$$support(A \Rightarrow B) = P(A \cup B)$$
$$confidence(A \Rightarrow B) = P(B \mid A)$$

The problem of finding the Association Rules can be divided into two parts [2]:

1. **Find all frequent item sets**: Frequent item sets will occur at least as frequently as a pre-determined minimum support count i.e. they must satisfy the minimum support [2].
2. **Generate strong association rules from the frequent item sets**: These rules must satisfy minimum support and minimum confidence values [2].

II. Algorithms In Association rule mining

Association Rule Mining has attracted a lot of intention in research area of Data Mining and generation of association rules is completely dependent on finding Frequent Item sets. Various algorithms are available for this purpose.

A. AIS Algorithm

In [2], an algorithm for frequent item set generation is proposed. This is the first algorithm being available to us for Frequent Item set discovery. It is known as AIS algorithm.

In this algorithm candidate itemsets are generated and counted at run time as database is scanned. AIS algorithm makes multiple passes over the dataset for the frequent item set generation. The term which is used in this algorithm is Frontier Set. This set undergoes extension during the pass. In each pass, the support for certain itemsets is measured. These itemsets, called candidate itemsets, are derived from the tuples in the database and the itemsets contained in the frontier set [2][3][6].

Initially the frontier set consists of only one element, which is an empty set. At the end of a pass, the support for a candidate itemset is compared with minsupport threshold value to check out if it is a large itemset (frequent itemset). At the same time, it is determined if this itemset should be added to the frontier set for the next pass. Those itemsets are added to the Frontier set which were expected to be less frequent but became frequent (large for current pass). The algorithm terminates when the frontier set becomes empty[2][3].

B. Apriori Algorithm

Apriori algorithm is given by Agrawal. It is used to generate frequent itemsets from the database. The Apriori algorithm uses the Apriori principle, which says that the item set I containing item set (say) X is never large if item set X is not large [1][7] or All the non empty subset of frequent item set must be frequent also.

TABLE I
NOTATIONS BEING USED IN APRIORI ALGORITHM

k itemset	Any itemset which consist of k items.
C_k	Set of Candidate k itemsets
L_k	Set of large k itemsets (frequent k itemsets). These itemsets are derived for the candidate itemsets in each pass.

Based on this principle, the Apriori algorithm generates a set of candidate item sets whose lengths are (k+1) from the large k item sets and prune those candidates, which does not contain large subset. Then, for the rest candidates, only those candidates that satisfy the minimum support threshold (decided previously by the user) are taken to be large (k+1)-item sets. The Apriori generate item sets by using only the large item sets found in the previous pass, without considering the transactions.
Steps involved are:

- Generate the candidate 1-itemsets (C_1) and write their support counts during the first scan.
- Find the large 1-itemsets (L_1) from C_1 by eliminating all those candidates which does not satisfy the support criteria.
- Join the L_1 to form C2 and use Apriori principle and repeat until no frequent itemset is found.

C. Direct Hashing and Pruning

In [10] DHP algorithm has been described which utilizes the extra data structure i.e. Hash Bucket for candidate itemset generation. The DHP (Direct Hashing and Pruning) algorithm is an effective hash-based algorithm for the candidate set generation. The DHP algorithm consists of three steps. The first step is to get a set of large 1-Item-sets and constructs a hash table for 2-Item-sets. The second step generates the set of candidate Item-sets Ck, but it only adds the k-Item-set into Ck if that k-Item-set is hashed into a hash entry whose value is greater than or equal to the minimum transaction support. The third part is essentially the same as the second part except it does not use the hash table in determining whether to include a particular Item-set into the candidate Item-sets.

TABLE II
DHP ALGORITHM PARAMETERS

Technique	Using Hashing Technique for finding large itemsets.
Time	Execution time is less consumed than Apriori algorithm for small databases.
Storage Structure	Array based technique is used.

D. Partition Algorithm

In [10][3] Partition Algorithm is discussed. It is variation of Apriori algorithm. In Apriori and DHP, there is problem of repeated passes on the database. In contrast Partition algorithm is composed of two passes on the database. The Partition algorithm logically partitions the database D into n partitions, and only reads the entire database at most two times to generate the association rules.

TABLE III
PARTITION ALGORITHM PARAMETERS

Technique	Partition the dataset to achieve local frequent itemset and hence finding global frequent itemset form them
Time	Execution time is bit more because of local frequent itemset generation
Storage Structure	Arrays are used generally.

III. IMPROVED TECHNIQUES FOR APRIORI ALGORITHM

Classical Apriori algorithm suffers from various drawbacks. Few of them are large candidate item set generation leading to high I/O cost , large number of scams on the database to find the support count in each pass.

Ways to improve the Apriori Algorithm[7]:

1. Reduce the number of transactions in the database.
2. Reduce the number of scans on the database.
3. Cut off the large candidates which cause high I/O cost.

The approach given in [4] provided a way to improve efficiency and reducing complexity. Firstly, database scanning is performed only once and further temporary tables have been used for scanning purpose. Secondly, it uses logarithmic decoding technique for reducing the complexity. Scheme provides good performance.

In [9], a method has been proposed to improve the efficiency of Apriori Algorithm using Transaction Reduction. Typical Apriori Algorithm generates a large set of Candidate sets if database is large. This method reduces the size of database which leads to reduced I/O cost. An attribute 'Size of Transaction (SOT) has been used which stores number of items in individual transaction in database. Transaction will be deleted according to the values of k (Pass number). Whatever will be the value of k, algorithm searches for the same value in database and when K= SOT, delete that particular transaction. In this way performance of Apriori algorithm has been improved by mining associations from massive data faster and better.

In [8], a new approach has been given for reducing the candidate item sets by reducing the connecting item sets i.e. frequent item sets of previous pass. In Classical Apriori algorithm:

- C_{k-1} is compared against support value.
- Item sets whose support count is less than support value (set by the user) will be eliminated /pruned and L_{k-1} will come out.
- L_{k-1} is connected with itself to form C_k.

The optimized algorithm is given by:

- C_{k-1} is compared against support value.
- Item sets less than support value will be eliminated/ pruned and L_{k-1} will come out.
- Before C_k will be generated, L_{k-1} will be further pruned on the basis of count the times of all the items occurred in L_{k-1} and delete those item sets with this number less than k-1 in L_{k-1}.

For large database, this optimized algorithm can save cost as well as time and hence increase the efficiency than the Apriori algorithm [8].

In [5], an approach has been suggested by using Bottom up approach along with using matrix and reduced transactions. The proposed algorithm composed of two phases i.e. Probability Matrix Generation and bottom up approach to find large item sets. Phases of the algorithm are:

Phase1: For the given dataset, an Initial matrix M1 will be generated for. Rows will represent transaction and Columns in matrix represent items. Each position in matrix will be having the value either 0 or 1which represents the presence or absence of item in particular transaction respectively.

From M1 Probability matrix M2 will be generated, where entry value of 1will is replaced by the probability of occurrence of corresponding item to the total number of transactions and two more columns will be added to the M2 which will store the total probability and count of elements in each row respectively. Then M2 will be arranged in descending order of Total Probability which leads to the formation of Sorted Probability Matrix M3 [5].

Phase 2: Non-zero entries in Sorted Probability Matrix M3 will be replaced by the value of 1 leads to the generation of Sorted Probability Matrix M4. Select first transaction from M4 and compare its total probability and count with next transaction total probability and count respectively.

If the current transaction total probability and count greater than the next transaction probability then perform the BITWISE AND operation between the transaction, if the resultant is equal to first transaction structure then increase the support count of first transaction item set by 1. Continue this process of Comparing and Bitwise AND operation with remaining transaction until it satisfies the condition of First transaction total probability and count is less than or equal to next transaction and checks the total support count if its greater than the required support count extract the item set of that transaction and all its subset and move it to frequent Item set until it finds unseen transaction in the given data set [5].

Hence proposed algorithm in [5] paper outperforms the Apriori Algorithm. Another advantage of this algorithm is once the largest frequent item set is found; all its subsets will be also moved to frequent items set. For next transaction, to find next largest frequent item set, first it checks whether item set of transaction under scanning process is already in frequent items Set or not. If present, then it will avoid scanning that transaction, hence leading to reduce complexity.

Hence on the basis of above approaches, these improvements have been compared on the basis of technique and benefits of each approach.

TABLE IV
COMPARISON OF IMPROVED VERSIONS OF APRIORI ALGORITHM

Authors	Technique	Benefit
Suhani Nagpal	-Temporary Tables for scanning. -Logarithmic Decoding	-Low system overhead and good operating performance [4]. -Efficiency higher than Apriori Algorithm.
Jaishree Singh, Hari Ram	Variable Size Of Transaction on the basis of which Transactions are reduced.	-Reduces the I/O cost. - Reduce the size of Candidate Item sets (Ck) [9].
Jaio Yabing	-Double Pruning method is used. -States that before Ck come out, prune Lk-1	For large datasets, it saves time and cost and increases the efficiency [8].

Sunil Kumar	–Probability Matrix has been used. -Uses Bottom Up approach.	Reduced Execution time than Apriori Algorithm [5].

IV. Conclusion

Various Classical Algorithms have been discussed in this paper like AIS, Apriori, Direct Hashing and Pruning and Partitioning algorithm. Different parameters has been discussed which are required for these algorithms to execute. Then methods to improve the Apriori Algorithm are mentioned and improved approaches have been discussed here also. A comparative study shows benefits of different approaches and technique used by these algorithms/ approaches.

V. References

[1] Rakesh Agrawal, Ramakrishnan Srikant, Fast Algorithms for Mining Association Rules, Proceedings of the 20th VLDB Conference Santiago, Chile, 1994.

[2] Rakesh Agrawal, Tomasz Imielinski, Arun Swami, Mining Association Rules between Sets of Items in Large Databases, Proceedings of the 1993 ACM SIGMOD Conference Washington DC, USA, May 1993.

[3] Anurag Choubey, Ravindra Patel, J.L.Rana, “A Survey of Efficient Algorithms and New Approach for Fast Discovery of Freqent itemset for Association Rule Mining”, IJSCE ,ISSN: 2231-2307, vol. 1, issue 2,May 2011.

[4] Suhani Nagpal, Improved Apriori Algorithm using logarithmic decoding and pruning, International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622, Vol. 2, Issue 3, May-Jun 2012, pp.2569-2572.

[5] Sunil Kumar , Shyam Karanth , Akshay K , Ananth Prabhu,Bharathraj Kumar M, Improved Apriori Algorithm Based on bottom upapproach using Probability and Matrix, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 3, March 2012,ISSN (Online): 1694-0814.

[6] R.Divya , S.Vinod kumar, Survey on AIS, Apriori and FP-Tree Algorithms,International Journal of Computer Science and Management Research,Vol 1 Issue 2 September 2012.

[7] Jiawei Han, Micheline Kamber. “Data Mining: Concepts and Techniques”, Morgan Kaufmann Publishers, Champaign:CS497JH, fall 2001.

[8] Jiao Yabing, “Research of an Improved Apriori Algorithm in Data Mining Association Rules”, International Journal of Computer and Communication Engineering, Vol. 2, No. 1, January 2013.

[9] Jaishree Singh, Hari Ram, Dr. J.S. Sodhi, ”Improving Efficiency of Apriori Algorithm using Transaction Reduction” International Journal of Scientific and Research Publications, Volume 3, Issue 1, January 2013 ISSN 2250-3153

[10] John D. Holt and Soon M. Chung, Efficient Mining of Association Rules in Text Databases, ACM 1999.

[11] Patel Tushar, Panchal Mayur, Ladumor Dhara, Kapadiya Jahanvi, Desai Piyusha, Prajapati Ashish, Prajapati, Research Journal of Computer and Information technology Sciences, Vol 1,2-5, February 2013.