Comparative Study of Different Improvements of Apriori Algorithm

Anneshya Ghosh
Department of Computer Science
Birla Institute of Technology, Mesra, Kolkata Campus
Kolkata ,India
anneshya.ghosh@gmail.com

Ambar Dutta
Department of Computer Science
Birla Institute of Technology, Mesra, Kolkata Campus
Kolkata ,India
adutta@bitmesra.ac.in

***Abstract*—** Data mining is a process of finding out the most frequent patterns from a large set of dataset. Association rule mining is an important technique in data mining. Apriori algorithm is the most basic, popular and simplest algorithm for finding out this frequent patterns. Still being one of the simplest algorithms for association rule mining, it has certain limitations. In the literature, researchers have proposed several improvements of Apriori algorithms. This paper provides the comparative study of some of these improved version of Apriori algorithm with respect to traditional Apriori algorithm.

Keywords- *Data Mining, Apriori algorithm, Frequent Itemset, Support, Dataset.*

*****

I. INTRODUCTION

Data Mining [1] is a process where information or knowledge is being extracted or mined from a large set of dataset. Here we mine the sequential patterns that are present in the large databases. Sequential Pattern Mining was first introduced by Agarwal and Srikant in the year 1995. Sequence patterns are those set of data which occurs in a specific order that is sequentially among all the data patterns in a given set. And finding out of these patterns which occurs sequentially out of all the other patterns is sequential pattern mining. An example of sequential pattern is as follows; suppose a customer buys a laptop then it is more likely that the customer will buy a mouse then antivirus and a printer after it sequentially. Some terms which are constantly being used here are item-set, support and confidence. Let there be a set of items L = {l1, l2, … } and a sub set of these items is knows as item-set. For a given database D, support of an item, let be X is defined as the ratio of the number of sequences in the database which contain the item X to the total number of sequences in the database. And, for a given database D, confidence of an sequence that contains X as well as Y is defined as the percentage of the number of sequences that contains X as well as Y to the number of sequences which contains X. Mining of sequential patterns can be classified into three different categories, they are as 1. Mining based on candidate generation (example, Apriori algorithm), 2. Mining without the involvement of any candidate generation (example, FP-Growth Tree algorithm) and 3. Mining item sets which have vertical format (example, ECLAT algorithm).

Mining of sequential patterns can be classified into three different categories, they are as 1. Mining based on candidate generation (example, Apriori algorithm), 2. Mining without the involvement of any candidate generation (example, FP-Growth Tree algorithm) and 3. Mining item sets which have vertical format (example, ECLAT algorithm).Apriori algorithm is the algorithm which involves Candidate generation. According to this algorithm, first the 1-itemsets are found then the database is scanned to find the support count. The itemsets with support count less than minimum support count are discarded. The resultant itemsets are then used to find the frequent 2-itemsets in the same process. Likewise we find all the (k+1)-itemsets from the frequent k-itemsets, until no more frequent itemsets can be found out. In FP-Growth tree algorithm [2], candidate keys are not generated and database is scanned for two times only. It uses a tree like structure to store the database and uses a divide and conquer method. And in ECLAT algorithm [2], depth first search method is used. In first scan of the database a TID (Transcation_Id) list is given to each single item. k+1 Itemset are then generated from the k itemset using apriori property and depth first search method. (k+1)-Itemset are then generated by taking the intersection of the TID-set of frequent k-Itemset. This process is continued, until no more candidates Itemset can be found.

In this paper, Apriori algorithm is taken into consideration. In section II, a detailed description of Apriori algorithm is provided along with its limitations. In section III, some of the existing improvements of Apriori algorithm are discussed with examples. In section IV comparisons between the original and the existing improved apriori algorithm is shown. Finally, conclusion is derived in section V.

II. APRIORI ALGORITHM

A. *Description*

The first and the most basic algorithm which was developed to find out the sequential patterns from a database was the Apriori algorithm [3]. This algorithm involves candidate generation and was first proposed by R. Agarwal and R. Srikant in the year 1994. In Apriori algorithm, we first scan the original database and find out the support count of each of the individual items. And discard those items whose support count is less than the minimum support count. The resultant item set is then used to find out the frequent 2-items set. From where again support count of each item-set is calculated and only those items whose support count is more than minimum support count are kept and others are discarded. Next we find out the frequent 3-item set and then frequent 4-item set until no more frequent item sets can be generated. The final frequent item set which is generated and satisfies the minimum support count is our final frequent pattern.

B. *Advantages and Disadvantages*

The advantage of Apriori algorithm is that it is a simple algorithm and can be implemented easily. But it still has some disadvantages also. The main disadvantage is that here the entire database needs to be scanned at each step. Also in this algorithm, a large number of candidate keys are generates. And if the database is very large than scanning in each step not only consumes a lot of time but the generation of a large number candidate keys consumes a lot of memory also, which can be sometimes limited3. Therefore this algorithm can work well for small database but not for large database.

III. Improvements of Apriori Algorithm

A number of improvements for Apriori algorithm have been proposed to overcome the limitations of the algorithm. In this section we will discuss some of the improvements and compare them. We will take into consideration those improvements which will reduce the number of scans and also the number of candidate key generation. The minimum support is taken as 3 for the database given below.

TABLE I. ORIGINAL DATABASE

Transaction	Items
T1	I1,I3,I7
T2	I2,I3,I7
T3	I1,I2,I3
T4	I2,I3
T5	I2,I3,I4,I5
T6	I2,I3
T7	I1,I2,I3,I4,I6
T8	I2,I3,I4,I6
T9	I1
T10	I1,I3

A. Algorithm for reducing the number of scan (Algorithm 1)

In this algorithm [4], we improve Apriori algorithm by reducing the number of scans. In this algorithm the first step is same as the classical Apriori algorithm. But in the second step, from each of the frequent 2-item set we first find out the one with minimum support count and then the transactions where that item is present. Next only from that transaction we check the frequent 2-items set and find the support count of individual set. For all the next frequent item set we do the same. In this we see the number of scans gets reduced. The algorithm works as shown in TABLE II, TABLE III and TABLE IV.

TABLE II. FREQUENT 1-ITEM SET

Items	Support	Transaction_IDs	
I1	5	T1,T3,T7,T9,T10	
I2	7	T2,T3,T4,T5,T6,T7,T8	
I3	9	T1,T2,T3,T4,T5,T6,T7,T8.T10	
I4	3	T5,T7,T8	
I5	1	T5	Deleted
I6	2	T7,T8	Deleted
I7	2	T1,T2	Deleted

From TABLE II, it is found that the number of scans for frequent 2–itemsets = (5+5+3+7+3+3) =26.

TABLE III. FREQUENT 2-ITEM SET

Items	Support	Item with Min_support	Transaction_IDs	
I1I2	2	I1	T1,T3,T7,T9,T10	Deleted
I1I3	4	I1	T1,T3,T7,T9,T10	
I1I4	1	I4	T5,T7,T8	Deleted
I2I3	7	I2	T2,T3,T4,T5,T6,T7,T8	
I2I4	3	I4	T5,T7,T8	
I3I4	3	I4	T5,T7,T8	

TABLE IV. FREQUENT 3-ITEM SET

Items	Support	Item with Min_support	Transaction_IDs	
I1I2I3	2	I1	T1,T3,T7,T9,T10	Deleted
I1I3I4	1	I4	T5,T7,T8	Deleted
I2I3I4	3	I4	T5,T7,T8	

The number of scans for frequent 3-itemset = (5+3+3) = 11, obtained from TABLE IV.

B. Algorithm for reducing database size and number of scans (Algorithm 2)

In this algorithm [5], Apriori algorithm was improved by both reducing the number of scans as well as cutting down the size of the database and removing some transactions which are not required. As a result the search time gets reduced by two times. In this algorithm also the first step is same as the original Apriori algorithm. And the for frequent 2-item set, we first delete all the transactions from the database which has less than 2 items, then for each individual 2-item sets we first find the item with minimum support among the two then search for the 2-ietms sets only in those transactions where the minimum support items are present and calculate their support count. We remove those item-sets which support count less than minimum support count. From the resultant set, frequent 3-item set s are obtained and so on. The algorithm is illustrated with the help of dataset provided in TABLE I. The improvements are shown in the tables – TABLE II, TABLE V, TABLE VI, TABLE VII and TABLE VIII.

TABLE V. REDUCED DATABASE FOR FREQUENT 2-ITEM SET

Transaction	Items
T1	I1,I3,I7
T2	I2,I3,I7
T3	I1,I2,I3
T4	I2,I3
T5	I2,I3,I4,I5
T6	I2,I3
T7	I1,I2,I3,I4,I6
T8	I2,I3,I4,I6
~~T9~~	~~I1~~
T10	I1,I3

TABLE VI. FREQUENT 2-ITEM SET

Items	Support	Item with Min_support	Transaction_IDs	
I1I2	2	I1	T1,T3,T7, T10	Deleted
I1I3	4	I1	T1,T3,T7, T10	
I1I4	1	I4	T5,T7,T8	Deleted
I2I3	7	I2	T2,T3,T4,T5,T6,T7,T8	

I2I4	3	I4	T5,T7,T8
I3I4	3	I4	T5,T7,T8

The number of scans for frequent 2–itemsets = (4+4+3+7+3+3) = 24.

TABLE VII. REDUCED DATABASE FOR FREQUENT 3-ITEM SET

Transaction	Items
T1	I1,I3,I7
T2	I2,I3,I7
T3	I1,I2,I3
~~T4~~	~~I2,I3~~
T5	I2,I3,I4,I5
~~T6~~	~~I2,I3~~
T7	I1,I2,I3,I4,I6
T8	I2,I3,I4,I6
~~T9~~	~~I1~~
~~T10~~	~~I1,I3~~

TABLE VIII. FREQUENT 3-ITEM SET

Items	Support	Item with Min_support	Transaction_IDs	
I1I2I3	2	I1	T1,T3,T7	Deleted
I1I3I4	1	I4	T5,T7,T8	Deleted
I2I3I4	3	I4	T5,T7,T8	

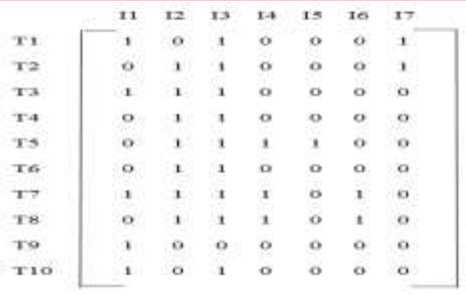
The number of scans for frequent 3-itemset = (3+3+3) = 9.

C. *Transaction Reduction and Matrix Method (Algorithm 3)*

In this proposed algorithm [6], items (I_n) and transactions (T_m) from the database are mapped into a matrix with size mxn. In this matrix, transactions are represented by the rows and the items are represented by the columns. The elements on this matrix are either 0 or 1 and is decides as follows:

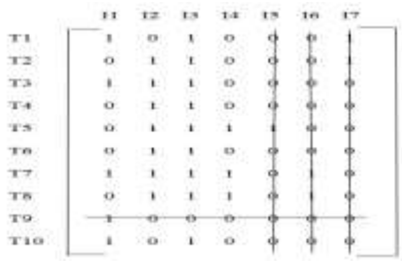
Martix = $[a_{ij}]$ = 1, if in transaction i there is item j, and
Matrix = $[a_{ij}]$ = 0, otherwise.

In the matrix, the sum of the row vector gives the sum of the transactions (S-O-T) and sum of the column vector gives the support count of each of the items. Now according to the algorithm, we first generate the items sets by the above 2 rules. Then for each frequent 1-item sets, we calculate the column vector and check if or not the value is more than the minimum support. If it is less, then the particular column is deleted. Then as it is frequent 1-item set we delete all rows whose row sum is equal to 1 or less than 1. Now from the resultant matrix we find out the frequent 2-item set by joining and deleting columns with column sum less than minimum support and row sum less than or equal to 2. Similarly we proceed for all the other frequent k-item sets by deleting columns with column sum less than minimum support and row sum less than or equal to k, until we find the frequent patterns. In this method the number of scans is reduced and also we don't have to check the whole database and has also reduced the I/O time spending in scanning database but still it some overhead as it has to maintain the updated database after each matrix generation. The algorithm is explained in Figure 1, Figure 2, Figure 3, Figure 4, Figure 5 and Figure 6.



	I1	I2	I3	I4	I5	I6	I7
T1	1	0	1	0	0	0	1
T2	0	1	1	0	0	0	1
T3	1	1	1	0	0	0	0
T4	0	1	1	0	0	0	0
T5	0	1	1	1	1	0	0
T6	0	1	1	0	0	0	0
T7	1	1	1	1	0	1	0
T8	0	1	1	1	0	1	0
T9	1	0	0	0	0	0	0
T10	1	0	1	0	0	0	0

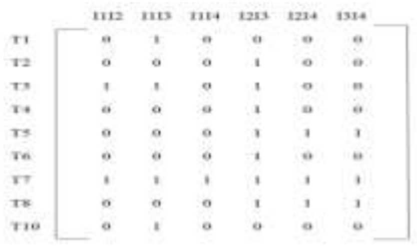
Figure 1. Frequent 1-itemsets.



	I1	I2	I3	I4	I5	I6	I7
T1	1	0	1	0	0	0	1
T2	0	1	1	0	0	0	1
T3	1	1	1	0	0	0	0
T4	0	1	1	0	0	0	0
T5	0	1	1	1	1	0	0
T6	0	1	1	0	0	0	0
T7	1	1	1	1	0	1	0
T8	0	1	1	1	0	1	0
T9	1	0	0	0	0	0	0
T10	1	0	1	0	0	0	0

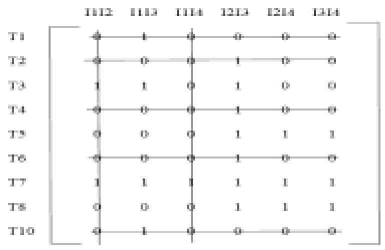
Figure 2. Frequent 1-Itemsets after deleting the rows and columns.

The number of scans for frequent 1-itemset = 10x7=70



	I1I2	I1I3	I1I4	I2I3	I2I4	I3I4
T1	0	1	0	0	0	0
T2	0	0	0	1	0	0
T3	1	1	0	1	0	0
T4	0	0	0	1	0	0
T5	0	0	0	1	1	1
T6	0	0	0	1	0	0
T7	1	1	1	1	1	1
T8	0	0	0	1	1	1
T10	0	1	0	0	0	0

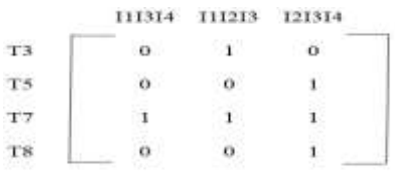
Figure 3. Frequent 2-itemsets.



	I1I2	I1I3	I1I4	I2I3	I2I4	I3I4
T1	0	1	0	0	0	0
T2	0	0	0	1	0	0
T3	1	1	0	1	0	0
T4	0	0	0	1	0	0
T5	0	0	0	1	1	1
T6	0	0	0	1	0	0
T7	1	1	1	1	1	1
T8	0	0	0	1	1	1
T10	0	1	0	0	0	0

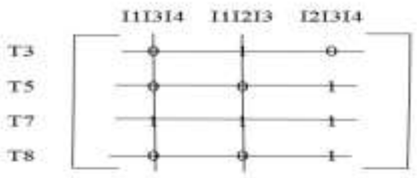
Figure 4. Frequent 2-Itemsets after deleting the rows and columns.

The number of scans for frequent 3-itemset = 9x6=54.



	I1I3I4	I1I2I3	I2I3I4
T3	0	1	0
T5	0	0	1
T7	1	1	1
T8	0	0	1

Figure 5. Frequent 3-itemsets.



	I1I3I4	I1I2I3	I2I3I4
T3	0	1	0
T5	0	0	1
T7	1	1	1
T8	0	0	1

Figure 6. Frequent 3-itemsets after deleting the rows and columns.

The number of scans for frequent 3-itemset = 4x3=12.

C. *Algorithm for reducing number of scans and candidate key formation (Algorithm 4)*

In this algorithm [7], we improve Apriori algorithm by reducing the number of scans and also the number of candidate keys that are formed at each step. Here the steps for finding out the frequent 1-item set and frequent 2-item set are same as the classical Apriori algorithm. The algorithm is improved from the next step where we find frequent 3-item set, frequent 4-item set and so on. For instance, in the example shown below the frequent 3-item sets that are formed by joining the frequent 2-item sets are {I1,I2,I3}, {I1,I3,I4} and {I2,I3,I4}. But we will not consider {I1,I2,I3} and {I1,I3,I4} as frequent 3-item sets because the individual patterns are not frequent, that is {I1,I2} and {I1,I4} are not frequent in each of the frequent 3-item set and {I2,I3,I4} is frequent 3-item set because each of the individual pair {I2,I3},{I3,I4} and {I2,I4} are frequent 2-item set. TABLE II, TABLE IX and TABLE X. Though the number of scans and candidate key generation is reduced in this improved version but still efficiency of the algorithm has not been improved at a high level.

TABLE IX. FREQUENT 2-ITEM SET

Item set	Support Count	
I1I2	2	Deleted
I1I3	4	
I1I4	1	Deleted
I2I3	7	
I2I4	3	
I3I4	3	

The number of candidate keys generated = 6.

TABLE X. FREQUENT 3-ITEM SET

Item set	Support Count
I2I3I4	3

The number of candidate keys generated = 1.

IV. Results and Discussion

In the above section we have discussed few algorithms developed to overcome the limitations of the Apriori algorithm. The result of all the algorithms are same, that is {I2, I3, I4}. And all the above algorithms have successfully reduced the number of scans. Some of them have also reduced the size of the database and candidate keys that are generated. In TABLE XI, we will compare the number of scans required in all of these algorithms. Hence we can conclude that the second algorithm has been able to reduce the number of scans to a huge extent. In TABLE XII, we compare the number of candidate keys generated. In this paper the comparison was made on a relatively small database. It can be seen that difference will be significant in the reduced values of the number of scans and the number of candidate keys generated on a larger database.

TABLE XI. COMPARISION AMONG THE NUMBER OF SCANS

Algorithm	Number of scans			
	1-Item set	***2-Item set***	***3-Item set***	***Total***
Normal Apriori algorithm	70	60	30	160
Algorithm 1	70	26	11	107
Algorithm 2	70	24	9	103
Algorithm 3	70	54	12	136
Algorithm 4	70	60	10	140

TABLE XII. COMPARISION AMONG THE NUMBER OF CANDIDATE KEYS GENERATED

Algorithm	Number of candidate keys			
	1-Item set	***2-Item set***	***3-Item set***	***Total***
Normal Apriori algorithm	7	6	3	16
Algorithm 4	7	6	1	14

V. Conclusions

The Apriori algorithm is the most basic algorithm that was developed for finding out the frequent patterns in large databases which suffered from certain limitations. In this paper, four proposals for the improvement of Apriori algorithm were discussed that have successfully overcome this limitations and their comparisons have been done and shown. But still these algorithms needs to be optimised more in terms of time consumption, memory requirement, efficiency and reduction in terms of number of scans.

References

[1] J. Han, M. Kamber, J. Pei "Data Mining Concepts and Techniques," Morgan Kaufmann Publisher, Third Edition, Year 2012.
[2] S. Nasreen, M. A. Azamb, K. Shehzada, U. N. M. Ali Ghazanfara, "Frequent Pattern Mining Algorithms for Finding Associated Frequent Patterns for Data Streams: A Survey" Procedia Computer Science 37, pages 109 – 116, Year 2014.
[3] R. Agrawal and R. Srikant. "*Fast algorithms for mining association rules*," In Proc. 1994 Int. Conf. Very Large Data Bases, pages 487–499, Santiago, Chile, September 1994.
[4] M. Al-Maolegi1, B. Arkok, AN IMPROVED APRIORI ALGORITHM FOR ASSOCIATION RULES," International Journal on Natural Language Computing (IJNLC), vol. 3, No.1, February 2014.
[5] S. Aggarwal and R. Sindhu, An approach to improve the efficiency of apriori algorithm. PeerJ PrePrints 3:e1410, 2015
[6] V. Mangla, C. Sarda, S. Madra, "Improving the efficiency of Apriori Algorithm in Data Mining," International Journal of Engineering and Innovative Technology (IJEIT), Vol. 3, Issue 3, September 2013.
[7] X. Fang, An Imroved Apriori Algorithm on the Frequent Itemsets, International Conference on Education Technology and Information System, Sanya, China, 845 – 848, 2013.