IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-8, Aug 2018 E-ISSN: 2347-2693

Improvisation in Efficiency of Apriori Algorithm for Mining Frequent Itemsets

D. Datta[1*], M.P. Dutta[1], R. Mukherjee[2]

[1]Department of Computer Science, St. Xavier's College (Autonomous), Kolkata, India
[2] A.K. Choudhury School of Information Technology, University of Calcutta, Kolkata, India

[*]*Corresponding Author: debabrata.datta@sxccal.edu*

Available online at: www.ijcseonline.org

Accepted: 13/Aug/2018, Published: 31/Aug/2018

Abstract— Association rule mining is a procedure which is meant to find frequent patterns from data sets found in various kinds of databases such as relational databases, transactional databases, etc. It has a great importance in data mining. Extracting relevant information from a huge collection of data by exploitation of data is called data mining. There is an increasing need of data mining by business people to extract valid and useful information from large datasets. Thus, data mining has its importance to discover hidden patterns from huge data stored in databases as well as data warehouse. Apriori algorithm has been one of the key algorithms in association rule mining. Classical Apriori algorithm is inefficient as it takes considerable amount of time to generate the desired output for mining the frequent itemsets owing to multiple scans on the database. In this research paper, a method has been proposed to improve the efficiency of Apriori algorithm by reducing the size of the database as well as reducing the time complexity for scanning the transactions.

Keywords—Itemsets, Apriori algorithm, Association rule mining, Minimum support

I. INTRODUCTION

Apriori algorithm proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database [2]. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database [12]. This algorithm has applications in domains related to market basket analysis. Data volumes are dramatically increasing by day-to-day activities, especially in the areas involving e-commerce [7]. Therefore, mining the association rules from massive data is in the interest for many industries as these rules help in decision-making processes, market basket analysis and cross marketing etc.

The algorithm proposed in this research paper in order to improve the efficiency of existing Apriori algorithm would be of considerable help in applications which require mining of huge datasets. It aims to overcome the following limitations of the traditional Apriori algorithm:

- Handling of large number of candidate and frequent itemsets resulting in increased cost and wastage of time
- Inefficiency in terms of memory requirement when large numbers of transactions are in consideration

The main objectives are to obtain quick outputs by optimizing the execution time, to reduce the search time, that is, the time taken to search for a data item by scanning the transactions and to reduce the time taken for generating the frequent itemsets. The improved algorithm would find its applications in various fields which involve dealing with huge collection of data like searching contents from the web etc. Second section discusses about some relevant work done in the concerned field. The proposed method has been put forward in the third section and the subsequent section has discussed about the results obtained. Finally, a concluding section has been added to make a point on the limitation of the proposed method and the corresponding scope of improvement.

II. RELATED WORK

Association rule problems have been in discussion from 1993 and many researchers have worked on it to optimize the original algorithm [1]. Many approaches have been proposed in past to improve Apriori algorithm but the core concept of the algorithm remained the same [3]. All the algorithms or methods used to improve the original Apriori algorithm have aimed at producing the same results which the original algorithm had produced but in an efficient and less cumbersome manner, consuming much less time. Another method which has been proposed for optimizing the algorithm was by designing a matrix based algorithm which had involved a bottom up approach and had improved the performance of the existing algorithm by reducing the

execution time but had involved an overhead to manage the new database after every matrix generation [8]. Another approach might be division of large database among processors [9]. A fourth technique called Double Pruning method had pruned L_{k-1} before C_k came out, where L_{k-1} was the large itemset containing (k-1) items and C_k was the candidate set having k items. For large datasets this method could save time and cost and also could increase the efficiency [14]. Yet another approach of improving the algorithm was by using temporary tables for scanning the transactions in the database. This methodology had involved logarithmic decoding, low system overhead and good operating performance and had efficiency higher than the existing Apriori algorithm [6]. Likewise there have been several other approaches which have been proposed to improve classical Apriori algorithm. In this paper, one method of improving the algorithm has been proposed and generating the candidate itemset in each iteration. The method has been discussed in the following section. The main focus on the newly proposed method has been on a reduced time for generation of the itemset. The fourth section has demonstrated the results obtained after applying the new method on a dataset.

III. PROPOSED METHOD

To improve Apriori algorithm efficiency, the proposed research work has given the main focus on reducing the time consumed for generating a candidate set, i.e., candidate-k (C_k) generation. In the process to find frequent itemsets, at first the size of a transaction (ST) has been found for each transaction in the database and has been maintained. Now, frequent-1 itemset (L_1) has been found and this set contains a of items, support value for each item and transaction ids containing the item. L_1 has been used to generate frequent-2 itemset (L_2), frequent-3 itemset (L_3) and so on along with decreasing the database size so that the time can reduce to scan the transaction from the database. To generate $C_k(x, y)$ (items in C_k are x and y), L_{k-1} has been joined with itself. To find L_2 from C_2 (candidate-2 itemset), instead of scanning complete database and all transactions, transactions with ST < k (where k is 2, 3,…) have been removed and also the deleted transactions from L_1 have been removed as well. This has helped in reducing the time to scan the infrequent transactions from the database. The minimum support from x and y has been found and the transaction ids of minimum support count item have been obtained from L_1. After that, C_k was scanned for specific transactions as mentioned and from the decreased database size. Then, L_2 has been generated by C_2 where support of C_k >= min_supp. Similarly, $C_3(x, y, z)$, L_3 and the subsequent sets have been generated repeating above steps until no frequent items sets can be discovered.

The process mentioned in the previous paragraph is depicted through a flowchart as given next:

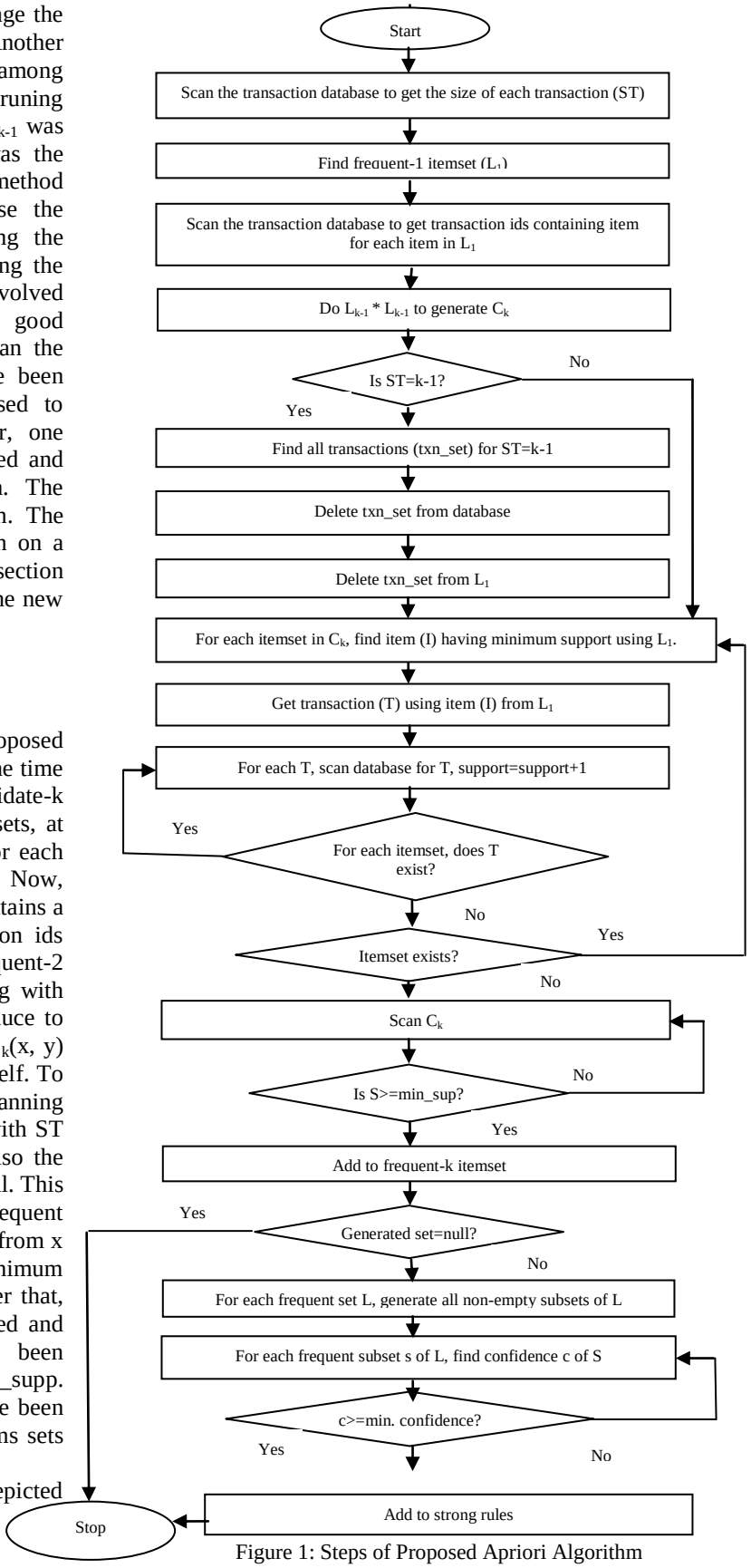

Figure 1: Steps of Proposed Apriori Algorithm

Accordingly, the improved Apriori algorithm has been designed and is stated below in the form of a pseudocode:

Algorithm Improved_Apriori
Input: transactions database, D
minimum support, min_sup

Output L_k: frequent k-itemsets in D
To find ST //for each transaction in database (DB)
L_1←find frequent_1_itemset (D)
L_1+=get_txn_ids(D)
for (k=2; L_{k-1}≠Φ; k++){
C_k=generate_candidate (L_{k-1})
x= item_min_sup(C_k, L_1) //find item from C_k(a, b) which has minimum support using L_1
target =get_txn_ids(x) //get transactions for each item
for each (txn t in tgt) do{
Increment C_k.count by 1
L_k=items in C_k which are greater than min_sup
} //end of for each
for each(txn in D){
if(ST=(k-1))
txn_set+=txn
//end of for
delete_txn_DB(txn_set) //reduce DB size
delete_txn_L_1(txn_set, L_1) //reduce transaction size in L_1
} //end for

A comparative analysis between the traditional Apriori algorithm and the one proposed in this paper shows that for generating frequent-1 itemsets, i.e., when k=1, both the algorithms scan equal number of transactions. However, as the value of k increases, the number of transactions scanned decreases in the case of the improved algorithm. In other words, for generating frequent-k itemsets (where k>=2) there occurs a significant difference in the number of transactions scanned by the two algorithms thus concluding that the proposed method is better than the existing one in terms of efficiency as it requires to scan much lesser transactions.

IV. RESULTS AND DISCUSSION

The proposed method has been implemented on a 32-bit machine having 4 GB RAM and Intel(R) Core(TM) i5-3210M as its processor with the processor speed being 2.50 GHz. The operating system used was Windows 7 Ultimate. The software tool used was Java using jdk 1.8.0_121.
To test the algorithm, a transaction table (D) having 10 transactions was used. Table 1 shows the transaction table. The minimum support value, min_sup was 3. The size of the transaction (ST) was calculated for each transaction and was shown in the table.

Table 1. Transaction table

Transaction	Items	ST
T_1	A, C, G	3
T_2	B, C, G	3
T_3	A, B, C	3
T_4	B, C	2
T_5	B, C, D, E	4
T_6	B, C	2
T_7	A, B, C,	5
T_8	B, C, D, F	4
T_9	A	1
T_{10}	A, C	2

All the transactions were scanned to get frequent 1-itemset, L_1 which contained items, respective support count and transactions from D. Infrequent candidates, that is, the itemsets whose support count values were less than the minimum support count value, min_sup were removed for further processing. These results are shown in the following two tables, table 2 and table 3.

Table 2. Candidate 1-itemset (C_1)

Item	Support
A	5
B	7
C	9
D	3
E	1
F	2
G	2

From table 2, it is observed that the last three rows contain items having support count value less than the required minimum supper value and hence they are not considered for any further processing.

Table 3. Frequent 1-itemset (L_1)

Item	Support	Transactions
A	5	T_1, T_3, T_7, T_9, T_{10}
B	7	T_2, T_3, T_4, T_5, T_6, T_7, T_8
C	9	T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8,
D	3	T_5, T_7, T_8
E	1	T_5
F	2	T_7, T_8
G	2	T_1, T_2

Accordingly, table 3 stores the frequent 1-itemset with the last three rows not considered for the next iteration.

From L_1, frequent-2-itemset (L_2) has been generated. The following example depicts the process. Considering itemset {A, B}, in classical Apriori algorithm, all the transactions are to be scanned to find {A, B} in D but in this proposed idea, at first, transaction T_9 is deleted from D as well as from L_1 as ST for T_9 is less than k (k=2). New D and L_1 are shown in table 4 and table 5 respectively. Secondly, {A, B} is split into {A} and {B} and item with minimum support, that is, {A} is selected using L_1 and its transactions will be used in L_2. So, {A, B} will be searched only in transactions which contain {A} i.e. T_1, T_3, T_7, T_{10}. Hence, the searching time is reduced in twofold way:

a) by reducing the database size

b) by cutting down the number of transactions to be scanned

Table 4. Updated transaction table

Transaction	Items	ST
T_1	A, C, G	3
T_2	B, C, G	3
T_3	A, B, C	3
T_4	B, C	2
T_5	B, C, D, E	4
T_6	B, C	2
T_7	A, B, C,	5
T_8	B, C, D, F	4
T_9	A	1
T_{10}	A, C	2

As stated before, from table 4, it is observed that T_9 would not be considered for further processing.

Table 5. Frequent 1-itemset

c	Support	Transactions
A	5	T_1, T_3, T_7, T_9, T_{10}
B	7	T_2, T_3, T_4, T_5, T_6, T_7, T_8
C	9	T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8,
D	3	T_5, T_7, T_8

For item A, T_9 is removed for the reason stated above.

From table 6, it is clear that the first row and the third row should be deleted as they don't satisfy the minimum support value condition.

Table 6. Updated Frequent 2-itemset

Item	Support	Min	Transactions
A, B	2	A	T1, T3, T7,
A, C	4	A	T1, T3, T7,
A, D	1	D	T5, T7, T8
B, C	7	B	T2, T3, T4, T5, T6, T7,
B, D	3	D	T5, T7, T8
C, D	3	D	T5, T7, T8

To generate frequent-3-itemset (L_3), D is updated by deleting transactions T_6 and T_{10} from table 4 as ST values for these transactions are less than k (k=3). L_1 is also updated accordingly by deleting transactions T_6 and T_{10}. Then, repeating above process, L_3 is generated and infrequent itemsets are deleted. Table 7, table 8 and table 9 show the updated database, L_1 and L_3 respectively.

Table 7. Updated transaction table

Transaction	Items	ST
T_1	A, C, G	3
T_2	B, C, G	3
T_3	A, B, C	3
T_4	B, C	2
T_5	B, C, D, E	4
T_7	A, B, C,	5
T_8	B, C, D, F	4

Table 8. Updated Frequent 1-itemset

Item	Support	Transactions
A	5	T_1, T_3, T_7
B	7	T_2, T_3, T_4, T_5, T_7, T_8
C	9	T_1, T_2, T_3, T_4, T_5, T_7, T_8
D	3	T_5, T_7, T_8

Table 9. Frequent 3-itemset

Item	Support	Min	Transactions
A, B, C	2	A	T_1, T_3, T_7
A, C, D	1	D	T_5, T_7, T_8
B, C, D	3	D	T_5, T_7, T_8

As described above, from table 9, the first two rows would be deleted and accordingly the final L3 would contain only itemset {B, C, D}.

In this way, the above method is followed to find a frequent k-itemset from any transaction list.

V. CONCLUSION AND FUTURE SCOPE

In this paper, the original or classical Apriori algorithm which has been used for mining frequent itemsets in a database has been modified to improve its performance efficiency. The modified version of the algorithm has given output at a faster rate and has involved less searching time and less number of scanning of the database.

The major objectives of designing the modified algorithm are to obtain quick outputs by minimizing the execution time, to reduce the search time and to reduce the time taken to generate the frequent itemsets.

The improved algorithm can be used in various technical fields to process large quantities of data in a quick and efficient manner. It can be used in internet search, digital advertisements, recommender systems, image recognition and various other fields.

Some of the proposed ideas related to the improvement of Apriori has been discussed in this research paper and the working of the improved algorithm has been explained by considering a randomly generated transaction database. The overall applicability of the proposed method can be further enforced if used with any real life transactions like in the field of e-commerce applications.

REFERENCES

[1] J. Han, M. Kamber, "Conception and Technology of Data Mining", China Machine Press, China, 2007.

[2] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, "From data mining to knowledge discovery in databases", Vol 17, Issue 3, AI magazine, pp. 37-54, 1996.

[3] S. Rao, R. Gupta, "Implementing Improved Algorithm over APRIORI Data Mining Association Rule Algorithm", International Journal of Computer Science And Technology,Vol 3, Issue 1, pp. 489-493, 2012.

[4] H. H. O. Nasereddin,"Stream data mining", International Journal of Web Applications, Vol 1, Issue 4, pp. 183–190, 2009.

[5] M. Halkidi, "Quality assessment and uncertainty handling in data mining process", In Proceedings of EDBT Ph.D. Workshop, Germany, 2000.

[6] R. Agarwal, R. Srikant, "Fast Algorithm for mining association rules", In Proceedings of 20th VLDB Conference, pp 487-499, 1994.

[7] Sakshi Aggarwal, Ritu Sindhu, "An Approach of Improvisation in Efficiency of Apriori Algorithm", In Proceedings of International Journal of Computer and Communication System Engineering, Vol 2, Issue 5, pp. 659-664, 2015.

[8] S. Kumar , S. Karanth , A. Prabhu, and B. Kumar , "Improved Apriori Algorithm Based On Bottom Upapproach Using Probability And Matrix",IJCSI,2012.

[9] F. H. AL-Zawaidah, Y. H. Jbara, A. L. Marwan, "An Improved Algorithm for Mining Association rules in Large Databases",World of Computer Science and Information Technology, Vol 1, Issue 7, pp. 311-316, 2011.

[10] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, D. Steinberg, "Top 10 algorithms in data mining,", Knowledge and Information Systems, Vol 14, Issue 1, pp. 1–37, 2007.

[11] F. Crespo, R. Weber, "A methodology for dynamic data mining based on fuzzy clustering", Fuzzy Sets and Systems, Vol 150, Issue 2, pp. 267–284, 2005.

[12] R. Agrawal, T. Imielinski, A. Swami, "Mining association rules between sets of items in large database,", In Proceedings of ACM SIGMOD International Conference on Management of Data, Vol 22, Issue 2, pp. 207–216, 1993.

[13] R. Bhaskar, S. Laxman, A. Smith, A. Thakurta, " Discovering frequent patterns in sensitive data", In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, , pp. 503-512, 2010.

[14] B. Vo, M. Chi, H. C. Minh, "Fast Algorithm for Mining Generalized Association Rules", International Journal of Database Theory and Application, Vol 2, Issue 12, pp. 161–180, 1994.

Authors Profile

Mr. D Datta pursued Master of Technology from University of Calcutta, India and he is currently pursuing his Ph.D. in Technology from the same university. He is an Assistant Professor in the department of Computer Science, St. Xavier's College (Autonomous), Kolkata, India He is a life member of IETE. He has published more than 20 research papers in reputed international journals and conferences His main research work focuses on Data Analysis. He has more than 10 years of teaching experience and has more than 4 years of Research Experience.

Mr. M. P. Dutta pursued his B.Sc. in Computer Science from St. Xavier's College (Autonomous), Kolkata, India and is currently doing his Masters in Computer Science from the same institute.

Miss R. Mukherjee pursued her B.Sc. in Computer Science from St. Xavier's College (Autonomous), Kolkata, India and is currently doing her Master in Computer Application from University of Calcutta, Kolkata.

