D-Apriori: An Algorithm to Incorporate Dynamism in Apriori Algorithm

S. Bagga
M.Tech student in Department of Computer Science & Engineering
DIT, Dehradun (U.K.), India

N. Badal
Assistant Professor in the Department of Computer Science & Engineering at KNIT, Sultanpur (U.P.), India

ABSTRACT
Apriori algorithm mines the data from the large scale data warehouse using association rule mining. In this paper a new algorithm named as Dynamic Apriori (D-Apriori) algorithm is presented. The proposed D-Apriori algorithm incorporates the dynamism in classical Apriori for efficiently mining the frequent itemsets from a large scale database. With the help of experimental results, it is shown that the D-Apriori algorithm performs better than the existing Apriori algorithm with respect to execution time for the dynamic behavior of data itemset.

Keywords
Association rule mining, frequent itemset, frequent patterns, Apriori, D-Apriori.

1. INTRODUCTION
One of the most powerful technology used to help companies to focus on the important information from the collected data about the behavior of their customers is Data Mining [9,10,12,13]. Data mining contains data analysis tools to find patterns and relationship from large set of data.

For data mining many techniques have been introduced. One of the most important techniques is association rule mining [1]. Association rule mining is used to find the association between the objects. In association rule mining, frequent itemsets plays an important role to find the frequent patterns. Frequent pattern defines how often the objects occur together in the database.

The best association rule mining algorithm is Apriori algorithm [15]. Apriori algorithm is used to find the frequent itemset from the database. Still there is a limitation that existing Apriori algorithm does not support the dynamic nature of frequent itemset. Therefore, there is a requirement to incorporate dynamism in existing Apriori algorithm. This paper proposed the new algorithm coined as Dynamic Apriori (D-Apriori) algorithm to overcome the drawback of existing Apriori algorithm that is for the large set of database multiple scan of database is required.

In the next section background is presented followed by the new proposed algorithm and experimental result for the proposed algorithm.

2. BACKGROUND
The basic working of the Apriori algorithm is illustrated in this section. The limitations of existing Apriori algorithm are also mentioned in the same section.

Apriori algorithm is the first algorithm of association rule mining introduced by Agarwal R. et. al. in [1, 2]. Steps of classical Apriori algorithm are described as follows-

i. First step of the algorithm is to count the support of each item separately from the sample database presented in table I. Table 1 contains 2 column- transaction ID (TID) and items. Table 2 contains the support count of each item.

Table 1: Sample Database item

TID	ITEMS
1	{1,2,3,4}
2	{1,2,3,4,5}
3	{2,3,4}
4	{2,3,5}
5	{1,2,4}
6	{1,3,4}
7	{2,3,4,5}
8	{1,3,4,5}
9	{3,4,5}
10	{1,2,3,5}

Table 2: Support of each

ITEMS	SUPPORT
1	6
2	7
3	9
4	8
5	6

ii. Next step eliminates the items which have support count less than the minimum support. Let the minimum support count be 5. Now the list of 2-pairs of frequent itemsets is generated and is presented in table 3.

In table 2 all the items are frequent so all the items are used in this step.

Table 3: 2-pairs of Frequent Itemsets

ITEMS	SUPPORT
{1,2}	4
{1,3}	5
{1,4}	5
{1,5}	3
{2,3}	6
{2,4}	5
{2,5}	4
{3,4}	7
{3,5}	6
{4,5}	4

iii. In the next step the items having support count less than minimum support are discarded and are described in table 4. The items {1,2},{1,5},{2,5}and {4,5} are discarded from table 4.

Table 4: Frequent Itemsets

ITEMS	SUPPORT
{1,3}	5
{1,4}	5
{2,3}	6
{2,4}	5
{3,4}	7
{3,5}	6

iv. Now the list of 3-pairs of frequent itemset is generated and is represented in table 5.

Table 5: 3-pair of Frequent Itemsets

ITEMS	SUPPORT
{1,3,4}	4
{2,3,4}	4
{3,4,5}	4

v. In table 5, support count is less than minimum support for all the itemsets, so 3-pairs of frequent itemset are not required.

Existing Apriori algorithm is also implemented in matlab which is presented by Badal N. et. al. in [3].

Other then Apriori algorithm, there are other algorithms which are used for the mining of frequent itemsets such as Trie [5, 6, 7, 8, 14], FP-Growth [11] and VS-Apriori algorithm [4].

Limitations of existing work:
The classical Apriori algorithm works well for the static nature of data itemset. If at the time of execution a new data itemset is introduced then the user has to restart the Apriori algorithm again. Classical algorithm does not support dynamic nature of the Apriori algorithm. Therefore dynamic nature of the frequent itemset is the major drawback with the existing Apriori algorithm.

D-APRIORI Algorithm

To overcome the limitations of existing Apriori algorithm, the new algorithm D-Apriori is introduced in this section. In D-Apriori algorithm the transactions can be added to the existing database at run time.

In this algorithm, the user can enter the *n* number of queries and the attributes used in each query. The queries and the corresponding attributes are stored in the matrix form in matlab. Then the existing Apriori algorithm is applied on the matrix created.

Working of D-Apriori algorithm is represented with the help of flow chart in figure 3.1 and is described as follows-

Flow Chart

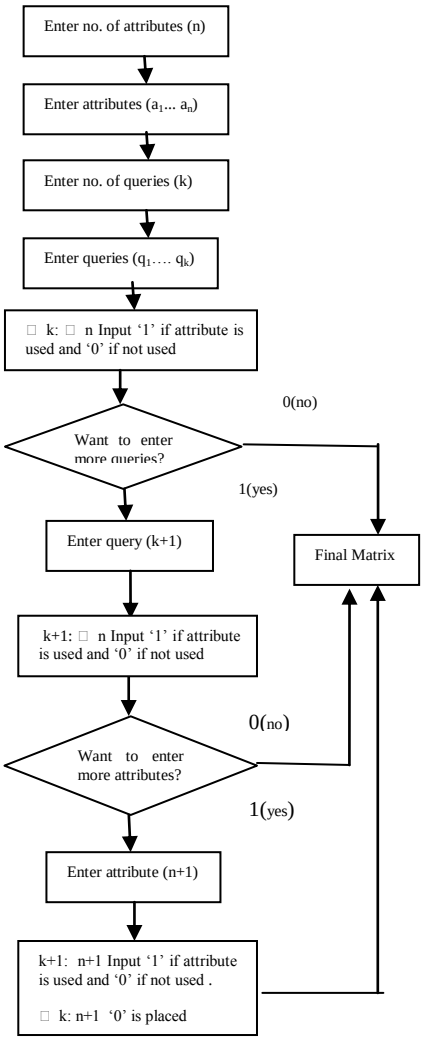

Figure 3.1: Flow Chart of D-Apriori Algorithm

First of all input number of attributes *n*. Then enter the names of *n* attributes a_1, a_2…..a_n . Next input the number of queries *k* followed by the names of *k* queries q_1, $q_{2...}$ q_k. Now input the value 1 if attribute is used in the query otherwise input 0. After this the matrix is created which will contain the values entered above. Then the each column of the matrix is multiplied with the other columns of the matrix including itself. These values are stored in the new matrix. Now the threshold value is entered and the sum of each column is compared with the threshold value. The sum of column less then threshold value is discarded. The columns which are repeated (example *a1a2 and a2a1*) and the columns which are multiplied to itself are also discarded. Finally the user gets the matrix which contains only frequent itemsets.

In D-Apriori algorithm, four cases occurs-
Case 1: No Queries No Attributes (NQNA)
In this case NQNA, if the user does not want to enter more queries and attributes then the final matrix is created with the existing attributes and queries which is similar to the existing Apriori algorithm.

Case 2: More Queries No Attributes (MQNA)
In the case of MQNA, if the user wants to enter more queries and no new attribute then the user has to input the

values '1' or '0' for the existing attributes. If '1' is entered then attribute is used otherwise attribute is not used.

Case 3: No Queries More Attributes (NQMA)
Case NQMA is not applicable because it is taken into consideration that user does not performs any error. For the existing queries, new attribute is entered only when the user has missed the particular attribute for the given queries.

Case 4: More Queries More Attributes (MQMA)
The last case MQMA is applicable only when the user wants to enter more attributes for the new query. The user has to input the value '1' or '0' only for the new attribute. For the existing attributes the value for all the query is placed '0' automatically. The user has not to enter this value.

Working of D-Apriori algorithm is illustrated with the help of example-

Input number of attributes- 2
Enter attributes-A1
Enter attributes-A2
Input number of queries-3
Enter queries-Q1
Enter queries-Q2
Enter queries-Q3
Input the value "1" if attribute is used, "0" if attribute is not used.
For query (1) (Q1) of attribute (1) (A1)-1
For query (1) (Q1) of attribute (2) (A2)-1
For query (2) (Q2) of attribute (1) (A1)-1
For query (2) (Q2) of attribute (2) (A2)-1
For query (3) (Q3) of attribute (1) (A1)-1
For query (3) (Q3) of attribute (2) (A2)-1
Want to enter more queries? [1] [0]-1
Enter queries-Q4
For query (4) (Q4) of attribute (1) (A1)-1
For query (4) (Q4) of attribute (2) (A2)-1
Want to enter more attributes for the same query? [1] [0] -1
Enter attributes-A3
For query (4) (Q4) of attribute (3) (A3)-1
The matrix formed from the above data is presented in figure 3.2-

	A1	A2	A3
Q1	1	1	0
Q2	1	1	0
Q3	1	1	0
Q4	1	1	1

Figure 3.2: Matrix containing attributes and queries

Now, each column of the matrix is multiplied and the result in stored in new matrix. Also the sum of each column is calculated which is presented in figure 3.3.

A3A3	A1A1	A1A2	A1A3	A2A1	A2A2	A2A3	A3A1	A3A2
Q1	1	0	1	1	0	0	0	0
Q2	1	0	1	1	0	0	0	0
Q3	1	0	1	1	0	0	0	0
Q4	1	1	1	1	1	1	1	1
SUM	4	1	4	4	1	1	1	1

Figure 3.3: Matrix containing multiplication of columns and sum

Enter the threshold value- 2

Now the column having the value of sum less than threshold value is discarded. The columns which are repeated (column A1A1, A2A1, A2A2, A3A1, A3A2 and A3A3) are also discarded.

Next matrix contains the frequent itemsets presented in figure 3.4.

	A1A2
Q1	1
Q2	1
Q3	1
Q4	1
SUM	4

Figure 3.4: Matrix containing frequent itemset

The frequent itemset is A1A2.

3 cases are generated from the above example-

Case 1: Figure 3.5 represents the final matrix created when No Queries No Attributes (NQNA) case occurs.

	A1	A2
Q1	1	1
Q2	1	1
Q3	1	1

Figure 3.5: Matrix containing existing queries and attributes

Case 2: In figure 3.6, final matrix is created for the case More Queries No Attributes (MQNA)

	A1	A2
Q1	1	1
Q2	1	1
Q3	1	1
Q4	1	1

Figure 3.6: Matrix containing new query and existing attributes

Case 3: Case (NQMA) is NOT APPLICABLE.

Case 4: Final matrix created for the case More Queries More Attributes (MQMA) is presented in figure 3.7.

	A1	A2	A3
Q1	1	1	0
Q2	1	1	0
Q3	1	1	0
Q4	1	1	1

Figure 3.7: Matrix containing new query and new attribute

Pseudo Code
The pseudo code of the proposed algorithm is presented.

Step1 is the candidate itemset of size k denoted by C_k.

Step 2 represents L_k, the frequent itemset of size k.

Step 3denotes the matrix A that contains the attributes of the database.

Step 4 represents the matrix Q that contains the queries of the database.

In step 5 matrix S contains the values of attributes occurred in each query.

Step 6 denotes the set of frequent itemset L1.

In step 7 L_k is incremented until it is equal to zero.C_{k+1} represents the candidate generated from L_k.

In step 8 the count of each candidate in C_{k+1} is incremented.

In step 9 L_{k+1} denotes the candidates with minimum support count that are in C_{k+1}.

Step 1. C_k: Candidate itemset of size k
Step 2. L_k : frequent itemset of size k
Step 3. A : Matrix containing attributes
Step 4. Q : Matrix containing queries
Step 5. S : Matrix containing the occurrence of attributes in each query
Step 6. L_1 = {frequent items};
Step 7. **for** (k = 1; L_k != Ø ; k++) **do begin**
C_{k+1} = candidates generated from L_k;
Step 8. **for each** transaction t in database do
increment the count of all candidates in C_{k+1} that are contained in t
Step 9. L_{k+1} = candidates in C_{k+1} with min_support
Step 10. **End**
Step 11. **return** $\hat{U}_k L_k$;

Figure 3.5: Pseudo Code of D-Apriori Algorithm

3. EXPERIMENTAL RESULT

The new D-Apriori algorithm is implemented in this section. D-Apriori algorithm is being compared with the existing Apriori algorithm with the help of sample database.

The proposed D-Apriori algorithm is implemented and compared on Intel(R) Core(TM) i3-2350M CPU@ 2.30GHz, 4GBand Matlab R2012a. Database connectivity is done using database toolbox from Matlab. The sample database containing 5000 transactions is taken to compare the execution time of D-Apriori algorithm and the existing Apriori algorithm.

In table 4.1, the comparison between the execution time and number of transactions is presented considering threshold value 2.

TABLE 4.1: Comparison between Apriori algorithm and D-Apriori algorithm

No. of Transactions	Apriori Algorithm (execution time in sec.)	D-Apriori Algorithm (execution time in sec.)
5	6.1	45.7
10	7.2	46.1
15	7.8	48.3
25	10.8	49.6
40	25.5	51.2
55	42.7	51.9
70	50.8	52.5
85	85.7	52.8
100	93.4	53.5
200	101.4	58.1
500	116.5	69.3
1000	149.9	84.7
3000	160.4	91.9
5000	178.7	105.6

Figure 4.1 is presented with the help of table 4.1. Table 4.1 represents the execution time (in sec) for D-Apriori algorithm and for existing Apriori algorithm.

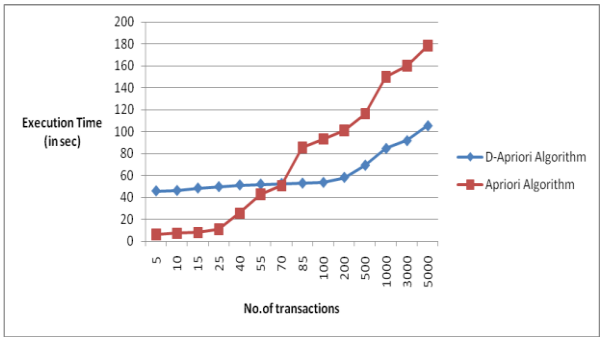

Figure 4.1: Comparison between implementationof D-Apriori algorithm and Apriori algorithm(execution time vs. number of transactions)

Figure 4.1 shows that the D-Apriori algorithm performs 74.64% better than the existing Apriori algorithm only for the large set of database.

4. CONCLUSION AND FUTURE SCOPE

The new presented D-Apriori algorithm improves the efficiency of existing Apriori algorithm for the large set of database. It is found that D-Apriori algorithm is best suitable for those data itemsets which are dynamic in nature. This algorithm reduces the execution time by 74.64% and is more reliable than classical Apriori algorithm only for the large set of database.

Further, work may be extended to introduce the intelligence with the selection of dynamic itemset as well as to initiate the algorithm.

5. REFERENCES

[1] Agrawal R., Imielinski T., and Swami A., "Mining Associations between seta of items in Massive Databases".In proc. Of the ACM-SIGMOD 1993 int'l conf. on Management of Data, Washingtom D.c USA, 1993A, pp. 207-216.

[2] Agrawal R., and Srikant R., "Fast Algorithms for Mining Association Rules", In Proc. VLDB 1994, pp.487-499.

[3] Badal N., Bagga S. ," Implementation of Apriori Algorithm in MATLAB using Attribute Affinity Matrix", International Journal of Advanced Research in Computer Science and Software Engineering", Vol. 4, No. 1, Jan. 2013, pp.10-15.

[4] Badal N. and Tripathi S., "Frequent Data Itemset Mining Using VS_AprioriAlgorithms",in International Journal on Computer Science and Engineering Vol. 02, No. 04, 2010, pp.1111-1118.

[5] Bentley J. L., "Multidimensional binary search trees used for associative searching" in International journal of Communications of the ACM, Vol. 19, 1975, pp. 509–517.

[6] Brin S., Motwani R., Ullman J. D., and S. Tsur, "Dynamic data itemset counting and implication rules for market basket data," SIGMOD Rec., Vol. 26, No. 2, 1997, pp. 255–264.

[7] Chan P. and Stolfo S., "Experiments in Multistrategy Learning by Meta-Learning", In Proc. of the second International conference on Information And Knowledge Management, 1993, pp. 314-323.

[8] Fredkin E.,"Trie memory". Commun. ACM, Vol. 3, No. 9, 1960, pp. 490–499.

[9] Gopalan N., Sivalselvan B.," Data mining Techniques and Trends", PHI Learning private limited, New Delhi 2009.

[10] Han J. and Kamber M.,"Data mining concepts and techniques", Elsevier, 2nd Edition. Chapter5.

[11] Han J., Pei J., and Yin Y., "Mining frequent patterns without candidate generation," in SIGMOD Conference, 2000, pp. 1–12.

[12] Ian H. and Frank E., "Data Mining: Practical machine learning tools and techniques", 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[13] Jerome H. Friedman. Data Mining and Statistics: What's the Connection? URL:http://stat.stanford.edu/~jhf/dm-stat.ps.Z

[14] Keleher P., Cox A. L., and Zwaenepoel W., "Lazy Release Consistency for Software Distributed Shared Memory". In Proc. of the 19th Annual Int'l Symposium on Computer Architecture, 1992, pp. 13-21.

[15] Liu X. W. and He P. L., "The research of improved association rules mining Apriori algorithm" Proceedings of

[16] 2004 International Conference on Machine Learning and

[17] Cybernetics, Vol. 3, No, 26-29 Aug. 2004, pp: 1577 – 1579.

6. AUTHOR'S PROFILE

N. Badal is an Assistant Professor in the Department of Computer Science & Engineering at KNIT, Sultanpur (U.P.), INDIA. He received B.E. (1997) from Bundelkhand Institute of Technology (BIET), Jhansi in Computer Science & Engineering, M.E. (2001) in Communication, Control and Networking from Madhav Institute of Technology and Science (MITS), Gwalior and PhD (2009) in Computer Science & Engineering from Motilal Nehru National Institute of Technology (MNNIT), Allahabad. He is Chartered Engineer (CE) from Institution of Engineers (IE), India. He is a Life Member of IE, IETE, ISTE and CSI-India. He has published about 35 papers in International/National Journals, conferences and seminars. His research interests are evinced at Distributed System, Parallel Processing, GIS, Data Warehouse & Data mining, Software engineering and Networking.

S. Bagga is an M.Tech student in the Department of Computer Science & Engineering at DIT, Dehradun (U.K.), INDIA. She received B.Tech (2011) from Lovely Professional University (LPU), Jalandhar in Information Technology.