IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

Trie Based Improved Apriori Algorithm to Generate Association Rules

Chirag Mewada[1], Rustom Morena[2]

Assistant Professor, Naran Lala College of Professional and Applied Sciences, Navsari, Gujarat, India[1]

Professor, Department of Computer Science, Veer Narmad South Gujarat University, Surat, Gujarat, India[2]

Abstract: The world is generating a massive amount of data. Therefore, there is a need for efficient methods to analyze and visualize enormous amount of valuable data being generated every day. Many methods are available for data mining, which extracts knowledge from data. But there is no method available which outperforms rest of them. Therefore, we have developed an algorithm based on a classic apriori algorithm and fp-growth algorithm to extract knowledge from data. We have used trie data structure to improve the performance by reducing the number of database scans. We have tested our algorithm on End of Day (EOD) data from November 2003 to August 2016 of Multi Commodity Exchange (MCX) of India. We found that our algorithm is faster than classic apriori algorithm.

Keywords: Improved Apriori Algorithm, Frequent Itemset Mining, Data Mining, Multi Commodity Exchange (MCX), Commodity Market

I. INTRODUCTION

With the increase of globalization and evolution of information technology, a massive amount of data are being generated therefore, there is a need for automated solutions for effective utilization of data to support decision making. [1] We have an enormous amount of valuable data available with us which we can analyze to discover some useful knowledge. This knowledge can be used for prediction or to better understand the overall process. A Huge amount of data is available in the form of terabytes which have drastically changed the areas of science and engineering. [2]

Data mining is science and technology for exploring data collected from various sources to discover previously unknown knowledge. Data mining algorithms can answer so many questions those traditionally are time-consuming to resolve. Data mining techniques can help us to reveal important data patterns that would otherwise remain unnoticed when using a simple type of analysis for massive amount of data available in data warehouses. [3] Data patterns exhibit some interesting facts about data which leads us to predict something that will be useful for decision making. Data mining and knowledge discovery applications have got a rich focus due to its significance in decision making and it has become an essential component in various organizations. [4] Various techniques are available to mine data like association rules, clustering, classification, regression, anomaly detection, decision tree, sequential pattern mining, etc.

For data mining, association rule mining is a most popular and well-researched method for discovering interesting relations between variables in large databases. [1] In association rule mining, we are actually trying to find interesting correlations between a large set of data items. From association rules, we can find out data items which occur together in the database. For example, if X occurs in the database along with Y many times then we can form a rule that if X occurs in the database then Y also occurs. If there are many items available in the database then there may be lots of rules and some of them may be useless. It is always difficult to select the appropriate data mining algorithm for specific database, there are many algorithms through which we can generate rules but it is always a problem to get rules with higher accuracy. [5] Therefore, interestingness measures are available to measure the quality of rules. One of them is confidence which is widely used measure to filter interesting rules from the whole set of rules. The best example of an association rule is market basket analysis. Market basket analysis provides frequent itemsets which express the customer's buying pattern. We can find out various combinations of products which are being purchased together by customers at supermarkets. The same concept can be applied to the stock market, derivatives market, banking, insurance, medical science, etc. There are so many algorithms available for association rule mining. Some of them are apriori, fp growth, eclat, aprioridp, context based association rule mining, node set based algorithms, etc.

Apriori is a classic algorithm for frequent item set mining and association rule learning from transactional databases. [6] Apriori algorithm was first proposed by R. Agrawal and R. Srikant. [7] Amongst all the other association rule mining algorithms, apriori can be used directly to generate association rules. Apriori algorithm is one of the most popular and widely used algorithms. It can be used to find frequent items from a transaction database. Apriori algorithm finds all sets of items which have support value more than the minimum support specified. To find

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

frequent itemsets, apriori algorithm scans transaction database multiple times. After the first scan, it finds individual frequent items. After each subsequent scan, it finds larger pairs of frequent items. It stops when largest itemsets are found. Apriori algorithm has two major limitations (1) It generates large candidate sets (2) It requires too many database scans. [8] It prunes unwanted items and itemsets.

Based on the support value, apriori algorithm removes non-frequent items and itemsets. Confidence is another interestingness measure which is used to form association rules from frequent itemsets found by using the apriori algorithm. Support is the total number of transactions where all items in A and B are together. Confidence determines how frequently items in B appear in transactions that contain A. [9] The formal definitions of both these metrics are given below,

$$\text{Support } (A \rightarrow B) = \sigma (A \text{ and } B)$$
$$\text{Confidence } (A \rightarrow B) = \sigma (A \text{ and } B) / \sigma (A)$$

II. LITERATURE REVIEW

M. J. Zaki, S. Parthasarathy, M. Ogihara, and W . Li [10] have proposed an algorithm for frequent itemset mining which scans the database only once. They have used clustering techniques to approximate the set of potentially maximal frequent itemsets. Their algorithm uses lattice traversal techniques to generate frequent itemsets contained in each cluster. They have used vertical database layout to cluster related transactions together. They first cluster itemsets using equivalence classes. Then they generate itemsets from each cluster using bottom-up traversal. They claim that their algorithm is better than the previously known algorithms for frequent itemset mining.

Jiawei Han, Jian Pei and Runying Mao [11] have proposed an algorithm to mine frequent itemset without generating candidates. They have used FP-tree which is an extended prefix tree structure. They have used FP-tree to store complex and crucial information of frequent patterns. They have developed FP-growth method which is based on FP-tree. By using this method they have reduced costly and repeated database scans.

They claim that their method is more efficient than not only apriori algorithm but other frequent pattern mining methods too. Despite being oldest and popular algorithm for mining association rules, FP-Tree is difficult to be used in an interactive mining system. [12]

Huan Wu, Zhigang Lu, Lin Pan and Rongsheng Xu [13] have proposed Apriori-based Algorithm (IAA). To reduce data scan they have used generation record. They have used a new count-based method to prune candidate itemsets. Their algorithm improves prune operation by using a count-based method; the count occurrence operation is improved by decreasing the scan data using generation record. They claim that their algorithm outperforms the original Apriori and some other existing Association Rule Mining (ARM) algorithms.

X. Luo and W. Wang [14] have proposed an algorithm to improve the apriori algorithm. In algorithm first, they make a Matrix library. The matrix library contains a binary representation where 1 indicates item present in transaction and 0 indicates it is absent. By counting the number of 1's in the matrix they find the occurrence of an item. For 2-itemset they multiply the binary representation of the items to get the occurrence of items together.

Support of two items can be calculated by a dividing number of times they appear together by total transactions. Similarly, the same procedure was followed for all possible itemsets. Their algorithm needs to scan the database only once and also does not require finding the candidate set when searching for frequent itemsets.

Abhijit Sarkar, Apurba Paul, Sainik Kumar Mahata and Deepak Kumar [15] have proposed a new algorithm for segregating data. They have modified the traditional Apriori algorithm. The amount of space required to store the data is considerably reduced by their approach. According to their method, first, they find 1 itemset and then they find frequent items. They construct a tree using the 1-itemset. The root node of a tree contains frequent itemsets which were derived from the 1 itemset.

Then, they create child nodes using frequent items found in the root node. They have used the formula level=n-1 to find the level of a tree, where n is the number of items in the root node of the tree. They create child nodes using all possible combination of (n-1) itemsets. Using bottom-up approach, they traverse the tree. They reject the parent if its child is infrequent. They claim that their algorithm outperforms apriori algorithm.

Jaishree Singh, Hari Ram and Dr. J.S. Sodhi [16] have proposed an Improved Apriori algorithm which cuts down unwanted transaction records to reduce scanning time. During pruning, they reduce the redundant sub-items. They directly from a set of frequent items and eliminate infrequent candidates. In their proposed method, they have used an attribute Size_Of_Transaction (SOT). SOT is a number of items in the individual transaction.

Harpreet Singh and Renu Dhir [17] have proposed a method based on transactional matrix and transaction reduction for finding frequent itemsets more efficiently. To remove deficiencies of a classic apriori algorithm like the generation of a large number of candidate itemsets and scanning the database too many times, they have proposed Matrix Based Algorithm with Tags (MBAT) which finds the frequent itemsets directly from the transactional matrix which is generated from the database to generate association rules. They claim that their algorithm greatly reduces the number of candidate itemsets, mainly candidate 2-itemsets.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

III. PROPOSED ALGORITHM

In our algorithm, we have used trie data structure to store frequent patterns. Trie data structure is used to store and retrieve words of a dictionary. A trie is a rooted and labeled tree. The depth of the root node is 0. A node at depth d points to nodes at depth d+1. Each parent node points to child nodes. There is an edge or link between each pair of parent and child nodes. An example of a trie is shown in figure 1.

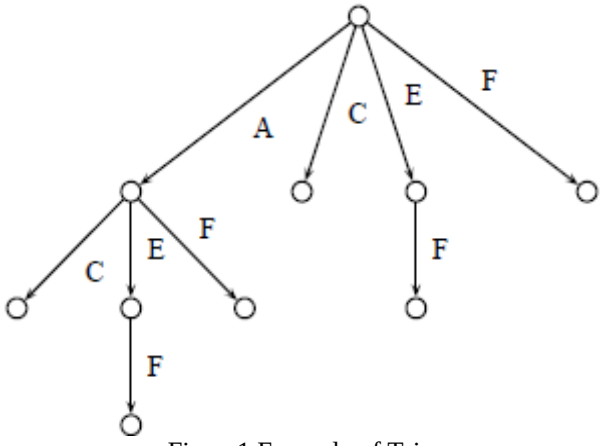

Figure1 Example of Trie

Tries are suitable for storing and retrieving not only words but any finite sets. [18] Therefore, we have used edges to store frequent items and nodes to store frequency count. Each edge of a tree contains a label and a link to the child node. Filtered transactions were stored into main memory and not on disk to reduce input output cost. In filtered transactions, we have eliminated infrequent items. Therefore, each filtered transaction contains only frequent items. We have also used a counter for repeated transactions to save memory and execution time. In APRIORI algorithm, collecting filtered transactions has a significant influence on run-time. [18] For counting support value, we took ordered transactions one by one. If we found a subset of the transaction in the trie, then we have increased the support count by a value which represents number of occurrences for a subset of the transaction. In our approach, trie stores not only candidates but frequent itemsets as well. After the first scan of the database, we have frequencies for each item. So to make search faster, we have used the order of frequency codes instead of actual items. The most frequent item was first then second frequent item and so on and so forth. Storing frequency codes and their inverses increase the memory need slightly, in return it increases the speed of retrieving the occurrence of the itemsets. [19] Frequency codes improve the speed of association rules generation but it slows down candidate generation. The difference of space requirement is negligible for candidate generation as compared to the improvement in speed of association rules generation. In FIM algorithms, tries are used to quickly determine the support of itemsets having size greater than two. [18] Therefore, we have used an array to count support for itemsets of size one and two. For itemsets having size more than two, we have used a trie to count support value. Classic APRIORI spends most of the time in determining the support of small and medium-sized candidates. In such cases, most edges lead to leaves hence removing other edges does not accelerate the algorithm too much. [18] Therefore, we have reduced pruning steps.

To improve the performance of our algorithm, we have performed pruning operation only in an array and not in a trie. Experiments show that memory need may be negligible to the third or the quarter. [18] We have created a trie which contains all possible combinations of sorted frequent itemsets. We do not scan transaction file to generate child nodes after itemsets of size 2.

Join Step: C_k is generated by joining $L_{k-1}$with itself
Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset

Pseudo-code of the algorithm is as follows:
T : Transaction database
R : Reduced transaction
ROT : Reduced and optimized transaction database
C_k: Candidate itemset of size k
L_k : Frequent itemset of size k
L_1 : Frequent items of size 1
j : Size of the candidate itemset
min_support : Minimum support
FindFrequentItems(k) : Function to find frequent itemset of size k

Improved Apriori Algorithm

Input: A transaction database T and a minimum support min_support
Output: U_kL_k the set of all frequent itemsets

```
1)    j = 1;
2)    L1 = CALL FindFrequentItems(j)
3)    FOR EACH transaction t in T
4)      FOR EACH item i in t
5)        IF L1 contains i THEN
6)          GENERATE Frequency Code Fk for i
7)          ADD Fk in R
8)        END IF
9)      END FOR
10)     IF ROT contains R THEN
11)       INCREMENT the count for R in ROT
12)     ELSE
13)       ADD R in ROT
14)     END IF
15)   END FOR
16)   WHILE Lk < > ∅
17)     j = j + 1;
18)     Uk Lk = CALL FindFrequentItems(j)
19)   END WHILE
```

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

FUNCTION FindFrequentItems(k)

1) IF k < > 1 THEN
2) TD = ROT
3) END IF
4) C_{k+1} = candidates generated from L_k
5) FOR EACH transaction t in TD
6) INCREMENT the count of all candidates in C_{k+1} that are contained in t
7) IF k < 3 THEN
8) L_{k+1}=candidates in C_{k+1} with min_support
9) ELSE
10) L_{k+1}= all candidates in C_{k+1}
11) End IF
12) END FOR
13) RETURN L_k
14) END FUNCTION

Above algorithm requires 3 times scan of transactions. For the first time, it scans transaction file and for the second and third time, it scans filtered transactions from memory which takes even less time than the previous scan. For itemset size 3 onwards, there are very few or may be no more non-frequent itemsets available in the candidate set. [18] Therefore, pruning operation is performed twice. After itemset size 2, pruning operation is not performed because most of the unwanted itemsets were removed during first two pruning operations.

This results in some unwanted memory waste but this can be compensated by improved performance. This algorithm follows a classic apriori algorithm to generate itemsets of size 1 and 2. From itemset size 3 onwards, we have modified the classic apriori algorithm to reduce the number of database scans which ultimately improves the performance. We have used reduced and transformed database. For frequent itemset of size 1, we are using original codes of each item contained in the transaction database. But after itemset size 1 we are creating frequency codes based on the frequency count value. An item with highest frequency count will be allocated frequency code 1 and second highest will be allocated 2 likewise rest of the frequent itemsets will be allocated frequency codes in order of their frequencies. From the original transaction database, we are creating reduced and optimized transaction file which contains frequency codes of only frequent itemsets. While creating optimized transactions in memory, we perform sorting on frequent items in each transaction. To make the search faster for counting support of frequent itemsets, we have sorted nodes in trie and items in transactions. Overall optimization takes little bit more time but that time will be compensated by the time saved while searching frequent items in a trie. Our algorithm generates more number of nodes as compared to classic apriori algorithm for itemset size 3 onwards because from itemset size 3 onwards numbers of unwanted itemsets are very less.

IV. COMPARATIVE ANALYSIS

TABLE I COMPARISON OF WIDELY USED FREQUENT ITEMSET MINING ALGORITHMS USING IMPORTANT PARAMETERS

PARAMETERS	**APRIORI** [19]	**FP GROWTH** [20]	**ECLAT** [21]	**IMPROVED APRIORI**
Number of Scans	Multiple	Two	One	One from transactions file and two from filtered transactions in memory
Storage Structure	Array and Tree	FP-Tree	Matrix	Array and Trie
Memory Requirement	Low	Average	High	Low
Running Time	High	Average	Less	Average
Search Type	Breadth First Search	Divide and Conquer Search	Depth First Search	Hybrid
Technique	Join and Prune	Conditional frequency pattern tree	Transaction lists intersection	Join and Prune
Database	Sparse/Dense	Large and medium	Small and Sparse	Sparse/Dense

V. EXPERIMENTAL RESULTS

In our experiment, we have used a system having 2.2GHz Core 2 Duo processor with 4GB main memory. We have used database populated from End of Day (EOD) price of future contracts of Multi Commodity Exchange of India (MCX). Sample of data collected from MCX for our experiment is shown in table II. We have pre-processed data to make it suitable for mining.

The pre-processed data contains 2450 instances and 27356 attributes. Dimensionality of our database is higher because multiple contracts for the same commodity are available for a particular day. Each attribute in database describes a combination of commodity name, expiry month and percentage change in commodity price as compared to the end of day price of the same contract of

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

the previous working day. Using apriori algorithm, we have tried to find correlated contracts of fundamentally different commodities.

From the result of an experimental study, it is clear that the performance of improved apriori algorithm is better than a classic apriori algorithm.

TABLE II SAMPLE DATA OF MULTI COMMODITY EXCHANGE (MCX)

Date	Commodity Name	Expiry Date	Close Price
02-05-2016	GOLD	03-02-2017	31173
02-05-2016	GOLD	05-12-2016	30873
02-05-2016	GOLDGUINEA	31-08-2016	24555
02-05-2016	GOLDGUINEA	29-07-2016	24490
02-05-2016	GOLDGUINEA	30-06-2016	24443
02-05-2016	GOLDGUINEA	31-05-2016	24409
02-05-2016	GOLDPETAL	31-08-2016	3043
02-05-2016	GOLDPETAL	29-07-2016	3020
02-05-2016	GOLDPETAL	30-06-2016	3016
02-05-2016	GOLDPETAL	31-05-2016	3010

From the study, we have noticed that as the support value decreases, the time taken by the apriori algorithm drastically increases. The time taken by improved apriori algorithm was always less than the classic apriori algorithm.

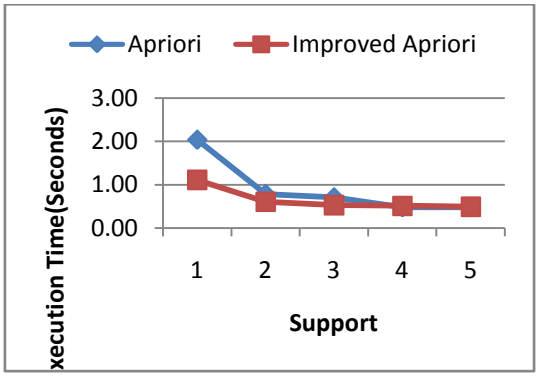

Figure 2 Execution time with increasing support value for future contract price data

Figure 2 is based on 12 years data of Multi Commodity Exchange (MCX) of India. It shows that when support value decreases, the execution time for both the algorithms increases.

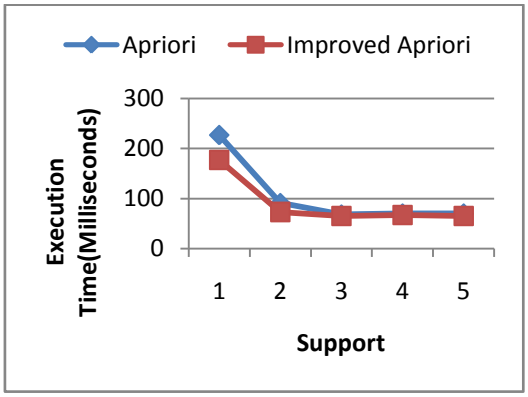

Figure 3 Execution time with increasing support value for one year future contract price data

The result also reveals the fact that the increase in execution time for improved apriori algorithm is always less than or equal to the time taken by the apriori algorithm. Figure 3 shows the trend for one-year data of Multi Commodity Exchange (MCX) of India. It shows that if support value increases then execution time decreases.
As we reduce support value, both algorithms take more time to mine data. Improved apriori requires less or equal time as compared to the classic apriori algorithm.

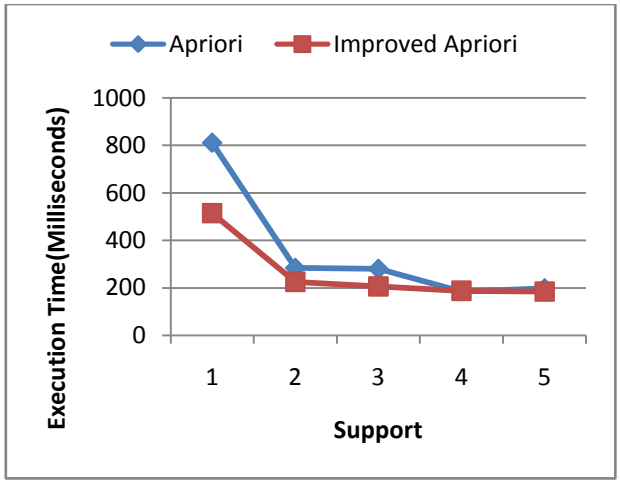

Figure 4 Execution time with increasing support value for three years future contract price data

Figure 4 is based on three years data of Multi Commodity Exchange (MCX) of India. It is based on more instances and attributes as compared to Figure 2. Still, both represent the common trend that as we reduce support value, improved apriori requires less or equal time than an apriori algorithm.

Figure 5 shows the result of an experiment done using six years future contracts of Multi Commodity Exchange (MCX) of India. The result portrays the same fact that we have seen in other figures based on various number of transactions.

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

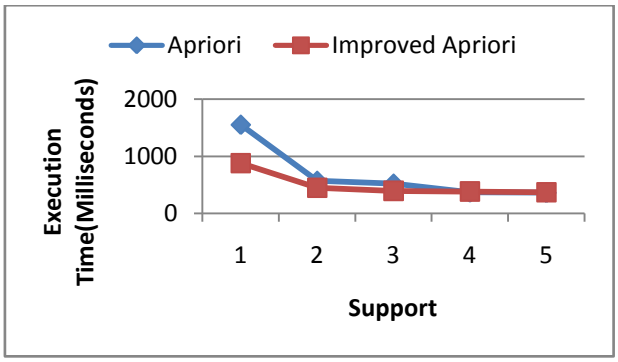

Figure 5 Execution time with increasing support value for six years future contract price data

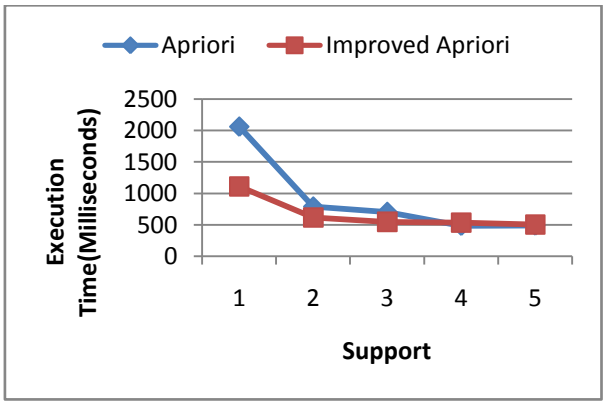

Figure 6 Execution time with increasing support value for nine years of future contract price data

Figure 6 shows the result which we have seen in figure 2 to figure 5. In our experiment with data of Multi Commodity Exchange (MCX) from November 2003 to August 2016, we got the result that improved version of apriori algorithm works faster than classic apriori algorithm. The result was confirmed from the experiments carried out by using data of one year, three years, six years and nine years. In our experimental study, we have used different numbers of transactions along with different support values. In all cases, our algorithm outperforms original apriori algorithm.

VI. CONCLUSION

Our algorithm is based on a classic apriori algorithm. We have tried to overcome limitations of the classic apriori algorithm. We have used trie data structure with filtered transactions. We have used reduced pruning technique to improve the performance of our algorithm. Our improved algorithm scans transaction file only once. From our experimental study, we can conclude that our algorithm is faster than the apriori algorithm to process transaction data regardless of the transaction database size.

REFERENCES

[1] E. Hajizadeh, H. D. Ardakani and J. Shahrabi, "Application of data mining techniques in stock markets: A survey," Journal of Economics and International Finance, vol. 2, no. 7, pp. 109-118, July 2010.

[2] N. Padhy, P. Mishra and R. Panigrahi, "The Survey of Data Mining Applications And Future Scope," International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), vol. 2, no. 3, June 2012.

[3] P. Desai and A. Desai, "The Study on Data Warehouse and Data Mining for Birth Registration System of the Surat City," in International Conference on Technology Systems and Management (ICTSM), 2011.

[4] B. K. Baradwaj and S. Pal, "Mining Educational Data to Analyze Students's Performance," (IJACSA) International Journal of Advanced Computer Science and Applications, vol. 2, no. 6, 2011.

[5] R. Thakkar and R. Morena, "A Performance Comparison between Rule Based and Association Rule Mining Algorithms in Extracting Knowledge from Stock Market Database," International Journal of Computer Science And Technology, vol. 5, no. 1, 2014.

[6] S. Patil and R. Deshmukh, "Review and Analysis of Apriori algorithm for Association Rule Mining," International Journal of Latest Trends in Engineering and Technology, vol. 6, no. 4, March 2016.

[7] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," in Proceedings of the 20th VLDB Conference, Chile, 1994.

[8] K. K. and R. Chezian, "A Survey on Association Rule Mining using Apriori Algorithm," International Journal of Computer Applications, vol. 45, no. 5, May 2012.

[9] K. Rajeswari, V. Vaithiyanathan, S. Tonge and R. Phalnikar, "Mining Association Rules using Hash Table," International Journal of Computer Applications, vol. 57, no. 8, p. 0975 – 8887, November 2012.

[10] M. J. Zaki, S. Parthasarathy, M. Ogihara and W. Li, "New Algorithms for Fast Discovery of Association Rules," in International Conference on Knowledge Discovery and Data Mining, Rochester NY, 1997.

[11] 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, vol. 8, pp. 53-87, 2004.

[12] Vivekananth.P, "Different Data Mining Algorithms: A Performance Analysis," International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), vol. 1, no. 3, 2012.

[13] H. Wu, Z. Lu, L. Pan and R. Xu, "An Improved Apriori-based Algorithm for Association Rules Mining," in Sixth International Conference on Fuzzy Systems and Knowledge Discovery, 2009.

[14] X. Luo and W. Wang, "Improved Algorithms Research for Association Rule Based on Matrix," in International Conference on Intelligent Computing and Cognitive Informatics, June 2010.

[15] A. Sarkar, A. Paul, S. K. Mahata and D. Kumar, "Modified Apriori Algorithm to find out Association Rules using Tree based Approach," in International Conference on Computing, Communication and Sensor Network(CCSN), 2012.

[16] J. Singh, H. Ram and J. S. Sodhi, "Improving Efficiency of Apriori Algorithm Using Transaction Reduction," International Journal of Scientific and Research Publications, vol. 3, no. 1, January 2013.

[17] H. Singh and R. Dhir, "A New Efficient Matrix Based Frequent Itemset Mining Algorithm with Tags," International Journal of Future Computer and Communication, vol. 2, no. 4, August 2013.

[18] F. Bodon, "Surprising results of trie-based FIM algorithms," in IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'04), Brighton, 2004.

[19] F. Bodon, "A fast APRIORI implementation," in IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03), Melbourne, Florida, USA, 2003.

[20] C. Borgelt, "An Implementation of the FP-growth Algorithm," in Workshop Open Source Data Mining Software ACM Press, New York, 2005.

[21] C. Borgelt, "Efficient Implementations of Apriori and Eclat," in Workshop of Frequent Item Set Mining Implementations, Melbourne, 2003.

[22] V. Mohan and D. S. Rajpoot, "Matrix-Over-Apriori: An Improvement Over Apriori Using Matrix," International Journal of Computer Science Engineering (IJCSE), vol. 5, no. 1, Jan 2016.

[23] A. Bhandari, A. Gupta and D. Das, "Improvised apriori algorithm using frequent pattern tree for real time applications in data mining," in International Conference on Information and Communication Technologies, 2014.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

BIOGRAPHIES

Chirag Mewada has obtained BCA and MCA degrees. He was working at Reliance Industries Ltd. in Mumbai from 2007 to 2009. He has industrial experience of two years in .net technologies and SAP ERP. He is associated with teaching in BCA since 2009. Presently, He is working as Assistant Professor at Naran Lala College of Professional and Applied Sciences, Navsari.

Rustom D. Morena has been associated with the teaching in MCA course since 1995. He has obtained BSc (Computer Science) and MCA degrees. He has been awarded MPhil in 2001 and PhD (Computer Science) in 2003. He has been teaching subjects such as Object Oriented Programming (C++), Client Server Architecture (ORACLE), System Development Tools (VB & VB.NET) and Analysis and Design of Algorithms. His areas of interest are data mining, database administration and financial analysis.