***I.J. Education and Management Engineering,* 2018, 6, 46-58**
Published Online November 2018 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2018.06.05

Available online at http://www.mecs-press.net/ijeme

Apriori Algorithm using Hashing for Frequent Itemsets Mining

Debabrata Datta [a,*], Atindriya De [b], Deborupa Roy [c], Soumodeep Dutta [d]

[a] *St. Xavier's College (Autonomous), Kolkata, India*
[b] *Jadavpur University, Kolkata, India*
[c] *Calcutta University, Kolkata, India*
[d] *Banaras Hindu University, Varanasi, India*

Received: 31 March 2018; Accepted: 06 August 2018; Published: 08 November 2018

Abstract

Data Warehousing, data mining and analysis plays a very important role in decision support. Various commercial organisations are using tools based on these techniques to be used for decision support system. Apriori algorithm is a classic algorithm which works on a set of data in the database and provides us with the set of most frequent itemsets. It is used to find the association rules and mines the most frequent itemsets in a set of transactions. Here the frequent subsets are extended one item at a time. In this paper a hash-based technique with Apriori algorithm has been designed to work on data analysis. Hashing helps in improving the spatial requirements as well as makes the process faster. The main purpose behind the work is to help in decision making. The user will select an item which he/she wishes to purchase, and his/her item selection is analysed to give him/her an option of two and three item sets. He/she can consider choosing a combination of two item sets or three item sets, or he/she can choose to go with his/her own purchase. Either ways, the algorithm helps him in making a decision.

Index Terms: Apriori algorithm, hashing, frequent itemsets, association rule, support count.

1. Introduction

Every year the amount of data being generated increases exponentially thus making the process of extracting useful information from them more tedious and critical. Most of these information are stored in a data repository known as the data warehouse. The data warehouse consists of data which are gathered from various sources inclusive of external sources, summarized form of information from the various internal systems as

* Corresponding author. Tel.:9830918949
E-mail address: debabrata.datta@sxccal.edu

well as from the corporate databases. Also storing the data is not enough, extracting useful information from this wide collection of data is equally important. This is where data analysis comes into play. It consists of writing simple queries, doing complex multidimensional analysis, data mining as well as presenting reports of statistical analysis.

A data warehouse consists of the data model, metadata as well as all the different rules in relation to data aggregation, distribution, replication, error handling and all the other various information required for the purpose of mapping the data warehouse. Data warehousing strategy mainly involves developing useful information from raw data using fast methods. The various data mining and data analysis tools make use of pattern recognition, cluster analysis, quantitative analysis, associations and correlation discovery for the purpose of data analysis without any kind of IT intervention [6]. There are two main ways of identifying frequent itemsets from a set of transactions:

Sequential: This process studies the order in which items are frequently purchased in a sequential data set, where sequence records an ordering of events.

Structured: This process searches for frequent sub-structures in a structured data set. Structures can consist of – graphs, lattices, sequence, sets, tree, single item or combination of above structures.

Each element of an itemset may contain a subsequence and such containment relationship can be defined recursively. Therefore, structural pattern mining can be considered as the most general form of frequent pattern mining. Once the useful information is extracted, it is presented to the users in a comprehensible form using processes which are collectively called Business Intelligence (BI). Managers can choose in between the various types of analysis tools available such as reports and queries, OLAP and its many variants (MOLAP, HOLAP and ROLAP). Data mining supports these and the patterns formed are later used for further data analysis thus completing the process of business intelligence [8]. Online Analytical Processing (OLAP) is one of the most popular data analysis technologies. Here, data are organized into cubes or multidimensional hierarchies using OLAP servers to enable fast analysis of data. The various data mining algorithms uncover patterns or relationships by scanning the databases. Data mining and OLAP are complementary where data mining takes up a bottom up approach for data analysis and OLAP provides the top-down approach. One such data analysis algorithm is known as Apriori algorithm.

With the escalation in the field of e-Commerce, data mining and analysis have seen an exponential growth. The aim is to find the hidden patterns for better understanding of customer buying style in retail sectors. For these problems Apriori algorithm is one of the most used solution for finding frequent itemsets from a transaction dataset and derive the corresponding association rule. This is a real-world example of the implementation of the Apriori algorithm. Based on the pattern of purchase found in the customers, the items which are most likely to be purchased by the customers are placed near each other to help him in decision making.

The present research work is enhancing the efficiency of the algorithm by implementing it with the hashing technique.

2. Related Work

There have been several works which have been done taking Apriori algorithm as the base method. One such research was described in [1] where Apriori Algorithm has been enhanced for discovering frequent patterns with an optimal number of scans. This particular paper has focused on the inadequacy of the Apriori algorithm for scanning the entire transactional database for the purpose of discovering association rules. The amount of time spent in scanning the entire transactional database is reduced by restricting the number of transactions while doing a calculation on the frequency of item-pairs or an item. Thus the improved algorithm optimizes the time complexity. Another research work, as depicted by [2], dealt with implementation of association rule mining using a reverse Apriori Algorithmic approach. This paper also made an improvement on the reverse Apriori and has compared the results with the existing classical one. A research work on parallel Apriori algorithm for mining frequent itemsets was described in [3]. This paper has revised Bodon’s implementation of finding the frequent itemsets into parallel ones where parallel computers were used to read the input transactions. They also presented the effect of the parallel computers on this implementation. Another research work [4] had proposed a new and enhanced form of the Apriori algorithm by presenting solution to the frequent set counting problem. They had optimized the beginning iterations of the Apriori, which consumes the maximum time when datasets with frequent length patterns of short or medium are considered. They had used a method of storing candidate items sets and keeping a count of their support and have also exploited the effective pruning methods, which has brought about a reduction in the size of the data set with the progress of the execution. For identifying frequent 2-itemsets, a research work was put forward in [5]. Another work has been done in [9] which illustrated Apriori algorithm on simulated database and found the association rule on different confidence value. Sheila A. Abaya [10] did related work in this field and used a modified Apriori algorithm which makes less database access and hence executes faster than the original algorithm. Use of Apriori algorithm has also been depicted in another research work [7]. Apriori algorithm has also been used for detecting suspicious emails [11]. Using deception theory which suggests reduced frequency of use of first person pronouns as well as exclusive words and more frequent usage of negative emotion words as well as action verbs point towards suspicious mails. Applying this model of deception on a set of email and after preprocessing the body of the email they have used Apriori algorithm that generates a classifier to categorize the email as deceptive or not.

3. Description of the Proposed Work

The main aim of the present research work is to generate a decision support system in a fast and efficient way. The proposed algorithm has been built based on the Apriori algorithm. It would identify a list of frequent item sets from the purchasing pattern coming from the users. This input to the proposed method has been taken in the form of a transaction string. Based on the various items present in the transaction string, the user will be given a choice of frequent item sets (one, two and three). The value of these frequent itemsets will be generated based on a particular parameter known as the support count.

For example, let there be a transaction table as shown in table 1. It consists of the transaction IDs and the corresponding transaction strings. The given table consists of four transactions. From T_1, the one-item candidate table (C_1) has been constructed and that has been shown in table 2. The candidate table contains all the items and their corresponding frequency values. Let the minimum support count be 2. Comparing the minimum support count and the frequency of the items in C_1, the frequent one item set table (L_1) is generated and has been shown in table 3 which comprises of items having frequency values greater than or equal to 2.

Table 1. Transaction table, T_1

Transaction Id	Itemset
T01	1,3,4
T02	2,3,5
T03	1,2,3,5
T04	2,5

Table 2. Table, C_1

Item	Support Count
1	2
2	3
3	3
4	1
5	3

Table 3. Table, L_1

Item	Support Count
1	2
2	3
3	3
5	3

Table 4. Table, C_2

Itemset
{1,2}
{1,3}
{1,5}
{2,3}
{2,5}

Table 5. Table C_2 with Support Count

Itemset	Support Count
{1,2}	1
{1,3}	2
{1,5}	1
{2,3}	2
{2,5}	3
{3,5}	2

A two item-set candidate table (C2) has been generated (table 4), which consists of item pairs (i, j) where $i < j$ and both i and j are from L1. The required item sets along with their support counts has been shown in table 5.

A frequent two item-set table (L_2) is now generated from the already constructed C_2 table. L_2 consists of all those item pairs (i, j) whose frequency in C_2 is greater than or equal to 2 (minimum support count). This has been shown in table 6.

Table 6. Table L_2

Itemset	Support Count
{1,3}	2
{2,3}	2
{2,5}	3
{3,5}	2

In a similar manner, a three item-set candidate set (C_3) and a frequent three itemset (L_3) may be generated. Now when a user selects a certain set of items to buy, the various combination of the items (1, 2 and 3) are formed and compared with the already generated frequent item-set tables (L_1, L_2, L_3). If the combination of items forms a match, then they are considered to be the most valued items.

Since the aim of the research work has been to incorporate a hashing function along with Apriori algorithm, hash tables have been generated to work with the items present in the tables containing the frequent itemsets. This was done to improve the efficiency of Apriori algorithm by considerably reducing the size of the candidate itemsets. A two itemset hash table (H_2) was used to store a combination of two items (y, x), where $y < x$, instead of the candidate table C_2. Each of these items in the item pair (y, x) was picked up from L_1 table. The hash function used for the purpose of storing the items was $[(y * 10) + x] \bmod 13$. A three-item set hash table (H_3) was used for the purpose of storing a combination of three items (z, y, x). Each of these items needs to be a part of L_1 table. It was also required that all possible two combinations of items z, y, x were belonging to the L_2 table and $z < y < x$ for all the items.

The hash function used for the purpose of storing the items in the hash table was $[(z * 100) + (y * 10) + x] \bmod 13$. To check the system integrity of the proposed work, different hash functions along with different sizes of hash tables were used. Various hash functions can be used for hashing and no hash function is collision free. The aim of using a particular hash function was to get minimal number of collisions. In the present research work, two different hash functions were used as depicted below:

1. Concatenation function – This function has concatenated the items present in 2 or 3 itemsets, as available, and then modulo operation was performed using a particular hash table size.
2. General Hash Function: A function has been defined and used. The function was $((z * 100) + (y * 10) + x) \bmod (\text{table size})$ for 3 itemsets combination and $((y * 10) + x) \bmod (\text{table size})$ for 2 itemsets combinations. Here z, y and x are index values of items and $z < y < x$.

The greater the hash table size the less has been the number of collisions. But at the same time, by increasing the hash table size, many spaces could be kept unused which would lead to a wastage of storage space. Hence a trade-off between number of collisions and space wastage has to be made while choosing the hash table size. In the proposed work, hash table sizes of 13 and 17 were used. Furthermore, the hash table size is mostly chosen to be a prime number as this leads to a comparatively less number of collisions.

4. Description of the Proposed Work

Step 1: Take the transaction string as input

Step 2: Update the 1-itemset candidate table C1 using the current transaction string by following these steps.

Step 2.a: Extract each item from the transaction string item list and update the count of that item in the C1 table.

Step 3: Items in C_1 with support count more than minimum support count, which are not present in frequent 1-itemset table L_1, are updated.

Step 4: All possible 2 item combination of the items present in the transaction string is generated using only those item pair both of which are present in L_1.

Step 5: The 2 item combinations generated in Step 4 are hashed into 2-itemset hash table H_2 using a hashing function.

Step 6: Frequent 2-itemset table L_2 is updated using the members of H_2 which has at least the minimum support count.

Step 7: All possible 3-item combination of the items present in the transaction string is generated using only those item sets where all the items are present in L_1 also all its 2-item combinations should be present in L_2.

Step 8: The 3 item combinations generated in Step 7 are hashed into 3-itemset hash table H_3 using some hash function.

Step 9: Frequent 3-itemset table L_3 is updated using the members of H_3 which has at least the minimum support count.

5. Result Analysis

The proposed method was executed in an environment with the following hardware and software platforms: Java SE 9.0.1 on Windows 10 operating system with Intel i5 5th generation processor.

Hash tables with two different sizes have been chosen for the implementation and the sizes were 13 and 17. Prime numbers have been chosen as the sizes of the hash table because of the fact that every key that has shared a common factor with the number of buckets would be hashed to a bucket that was a multiple of this factor. Therefore, the common factor between them needed to be reduced, so selecting a prime number as a mod in hash function has been the best way.

Table 7. Transaction Table

Id	Item list
1	5,6,10,50,48,10
2	15,26,48,5,10,20
3	34,26,5,25,28,31,1
4	5,1,26,48,4,3,42
5	6,29
6	19,46,37,17
7	28,17,39,50
8	16,28,17,38,5,24
9	5,22,33,44,11
10	28,15,6,28,8,41
11	25,26,10,50,48,10,11,1
12	15,36,48,5,10,20,1,2, 3
13	4,15,44,43,21,7
14	15,1,12,17,19,21,7
15	8,8,8,8,8,8,16,16,16
16	8
17	28,29,30,31, 32,27,39
18	35,45,7,6,5,10
19	7,17,23,35,49
20	18,28

Since two different hash functions have been used with two separate table sizes, the proposed method could be run with four different cases as mentioned below:

- using concatenation hash function and keeping hash table size 13
- using the general hash function and keeping hash table size 13
- using concatenation hash function and keeping the hash table size 17

- using the general hash function and keeping the table size 17

Table 8. 1-itemset Candidate Table

Item	Support Count
1	4
2	1
3	2
4	3
5	8
6	4
7	3
8	8
9	0
10	7
11	3
12	2
13	0
14	1
15	5
16	4
17	5
18	1
19	3
20	3
21	2
22	1
23	1
24	1
25	3
26	5
27	1
28	7
29	2
30	1
31	2
32	1
33	1
34	1
35	2
36	1
37	1
38	1
39	2
40	0
41	1
42	1
43	2
44	2
45	1
46	1
47	0
48	5
49	2
50	4

For the implementation purpose, twenty randomly generated transactions on fifty different items have been created. The items have been numbered from 1 to 50. Table 7 stores the information for the transaction data.

Each transaction row is available to us through a transaction string I1, I2, I3…. In. On every such transaction string the algorithm is executed, and the hash table is updated.

Corresponding 1-itemset candidate table C1, for fifty items has been generated as shown in table 8 and the corresponding frequent 1-itemset table L1 has been shown in table 9. Here we are considering 1 as the minimum support count. Both the tables have been shown in the next page.

After this, hash functions have been applied for different cases. The first case is depicted below:

The hash function that has been initially chosen is based on the concatenation operation. The function is first applied on 2-itemset pair. It concatenates each item of the 2-itemset pair and performs mod 13, the size of the hash table, which initially has been chosen. To generate the H_3 table another concatenation operation was performed on x, y and z and then the modulus was performed and the value we got as result is the hashing index for that itemset in the hash table. Here z, y and x are index values of items and $z < y < x$. The result of the execution of 2-itemsets has been shown as table H_2 and that of 3-itemsets as table H_3.

Table 9. Frequent 1-itemset table

Item	Support Count
1	4
5	8
6	4
8	8
10	7
15	5
16	4
17	5
26	5
28	7
48	5
50	4

Table 10. Few Entries of Table H_2

A	b	C	D	e	f	G
Null	null	Null	null	null	null	5, 26
Null	null	5, 48	null	null	null	5, 26
Null	null	Null	null	6, 28	null	Null
5, 20	48, 50	Null	null	null	null	Null
25, 48	15, 48	5, 48	5, 10	11, 48	5, 25	5, 26
Null	null	11, 20	null	15, 25	15, 26	10, 20
Null	null	Null	null	25, 26	null	Null
1, 17	null	1, 19	17, 19	null	null	Null
Null	null	Null	5, 10	5, 6	null	Null
Null	null	Null	null	null	null	5, 26

Table 10 shows the all the records stored in H_2. In this table, the column headings, denoted by characters starting from ‘a’, signify thirteen possible outcomes after applying the hash function as the modular operation has been performed on 13. Here ‘a’ stand for 0 and so on and so forth. Table 10 has shown the entries for the first seven outcomes, denoted from ‘a’ to ‘g’. From H_2, the frequent 2-itemset table (L_2) has been generated. Table 11 shows some of the entries from L_2.

Table 11. Few Entries of Table L_2

Item	Support Count
1,17	1
1,19	1
10,48	2
11,15	1
5,10	2
5,17	1
15, 17	1

In table 11, the support count of the itemset {1, 17} is 1 as this set has occurred only once in H_2. Similarly, the support count values of other itemsets have been depicted in L_2. From L_2, H_3 has been generated and some of the entries of H_3 have been shown as table 12.

Table 12. Few Entries of Table H_3

a	b	c	d	e	f	g
null	5,17,28	null	null	null	null	Null
10,26,48	26,48,50	10,26,50	null	null	10,48,50	Null
1,10,11	1,5,48	1,5,10	1,5,11	1,5,25	1,5,26	5,11,48
1,11,15	1,10,25	1,10,26	1,25,48	1,10,15	1,11,20	5,15,25
1,20,25	1,20,26	1,15,20	5,10,15	1,15,48	10,15,48	5,25,26
5,10,25	5,10,26	5,25,48	5,15,48	5,11,20	11,20,26	10,11,20
5,20,26	5,15,20	10,20,26	10,15,20	10,25,48	null	11,15,20
10,26,48	10,11,15	15,26,48	11,26,48	11,20,25	null	15,25,48
15,20,48	10,20,25	null	15,20,25	15,20,26	null	25,26,48
null	11,20,48	null	null	20,26,48	null	null
null	1,15,19	null	null	null	null	1,17,19

From H_3, the frequent 3-itemset table (L_3) has been generated. Table 13 shows some of the entries from table L_3.

Table 13. Few Entries of Table L_3

Item	Support Count
10,26,48	2
1,10,11	1
1,15,19	1
1,5,11	1
10,25,48	1
11,20,25	1
11,15,20	1

In table 13, the support count of the itemset {10, 26, 48} is 2 as this set has occurred twice in H_3. Similarly, the support count values of other itemsets have been depicted in L_3.

These hash tables once calculated, are stored in the database. When a customer makes a new transaction, he is given an option of 2 and 3 most frequently bought itemset with respect to the items in his transaction list, if those items have been bought quite frequently with other items. In the second case, another hash function has been applied on the set of transactions as mentioned in table 7.

This hash function is stated below:

a. ((y * 10) + x) mod 13 in case of 2-itemsets
b. ((z * 100) + (y * 10) + x) mod 13 in case of 3-itemsets

In both the cases, table size was taken to be 13 and z, y and x are index values of items and z < y < x. Table 14 depicts the entries of H_2, which stores the result of the execution of hash function on 2-itemsets and that on 3-itemsets has been shown as table H_3 and table 16 depicts the entries of H_3.

Table 14. Few Entries of Table H_2

a	b	c	d	e	f	g
5,28	null	5,17	17,28	null	null	null
null	null	null	null	null	10,48	null
5,15	15,20	1,5	10,20	1,20	5,20	1,48
11,20	20,48	11,48	15,48	20,25	10,48	11,26
null	null	null	25,26	null	11,25	15,25
null	null	null	null	null	20,26	null
null	null	null	4,15	null	null	null
15,19	1,17	null	1,19	null	null	null
null	null	null	null	null	8,16	null
null	null	null	null	5,6	6,10	null

As has been done in the first case, from H_2, the frequent 2-itemset table (L_2) has been generated. Table 15 shows some of the entries from table L_2.

Table 15. Few Entries of Table L_2

Item	Support Count
5, 28	1
15,19	1
1,17	1
25,26	1
1,5	1
4,15	1
1,19	1

In table 15, the support count of the itemset {5, 28} is 1 as this set has occurred only once in H_2. Similarly, the support count values of other itemsets have been depicted in L_2. From L_2, H_3 has been generated and some of the entries of H_3 have been shown as table 15.

Table 16. Few Entries of Table H_3

A	b	c	d	e	f	g
null	null	5,26,48	null	null	null	null
1,20,25	1,5,20	1,11,26	1,5,48	1,5,10	1,5,11	1,5,25
5,10,11	1,10,48	1,15,25	1,10,11	1,10,25	1,10,26	5,11,20
5,15,26	1,11,25	5,10,26	1,15,26	1,11,15	1,26,48	10,15,26

From H_3, the frequent 3-itemset table (L_3) has been generated. Table 17 shows some of the entries from table L_3. In table 17, the support count of the itemset {5, 26, 48} is 1 as this set has occurred once in H_3. Similarly, the support count values of other itemsets have been depicted in L_3.

Similarly, other two cases have been executed with table size 17 and the corresponding results were obtained as expected.

Table 17. Few Entries of Table L_3

Item	Support Count
5,26,48	1
1,20,25	1
1,15,48	1
1,15,26	1

Furthermore, the proposed algorithm doesn't depend upon the size of the transaction table in the sense that, if every time a new transaction takes place the whole transaction table won't be rescanned to generate the updated 1,2 and 3 frequent itemset table, instead the frequency of the new items purchased are updated in the hash table and frequent itemset table. This in turn saves time as the whole table is not scanned after every transaction which becomes a tedious job once the size of the transaction table increases as well as when the number of items in the inventory increases. Also, since we are using hashing to store the 2 and 3 itemsets, we don't need a candidate table an auxiliary data structure, which needs to be scanned completely every time to find the frequency of an itemset present in the table. This is achieved as same itemset appear under same index value and hence their count can be found by traversing that index location only, where the itemsets have been stored using chaining to resolve collision.

Thus, by implementing the original Apriori algorithm using hashing, time complexity could be decreased.

6. Conclusion and Future Scope

On comparing the H_2 table of size 17 and 13 for the general hash function, it has been observed that both are suffering quite less collisions but the H_2 of size 13 is more compact and contains less NULL entries than that of size 17. Thus, for this set of transaction H_2 of size 13 is preferred. But since the number of transactions may increase in practice, a hash table with bigger size would be preferred to reduce the collisions. In such a case, H_2 of size 13 can be rehashed in H_2 of size 17.

The same conclusion holds for hash table H_3 of size 13 and 17.

On comparing H_2 of size 13 for two different hash functions, it has been observed that there is no significant difference between the outcomes stored in two H_2 tables. The number if elements remain quite the same; only the hashed position has been different.

The same holds for H_3 table also. The same case is also true for hash table of size 17.

Further, a better hash function can be used so that the number of null positions may be reduced in the hash tables and to improve the spatial complexity of the algorithm further. Though finding a perfect hash function based upon transaction is difficult because the transactions are random and thus it is difficult to design a hash function based upon random transactions.

References

[1] Sudhir Tirumalasetty, Aruna Jadda, Sreenivasa and Reddy Edara, "An enhanced Apriori Algorithm for Discovering Frequent Patterns with Optimal Number of Scans", International Journal of Computer Science, 2015.

[2] Ashma Chawla and Kanwalvir Singh Dhindsa, "Implementation of Association Rule Mining using Reverse Apriori Algorithmic Approach", International Journal of Computer Applications, Vol. 93, No.8, 2014, pp. 24 – 28.

[3] Yanbin Ye and Chia-Chu Chiang, "A parallel Apriori Algorithm for Frequent Itemsets Mining", In proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications, 2006, pp. 87 – 94.

[4] Raffaele Perego, Salvatore Orlando and P.Palmerini, "Enhancing the Apriori Algorithm for Frequent Set Counting", In proceedings of the Third International Conference on Data Warehousing and Knowledge Discovery, 2001, pp. 71 – 82.

[5] K.Vanitha and R.Santhi, "Using Hash Based Apriori Algorithm To Reduce The Candidate 2- Itemsets For Mining Association Rule", Journal of Global Research in Computer Science ,Vol. 2, No. 5, 2011, pp. 78 – 80.

[6] Nick Roussopoulos, "Materialized Views and Data Warehouses", ACM SIGMOD Newsletter, Vol. 27, Issue 1, 1998, pp. 21 – 26.

[7] Shikha Bhardwaj, Preeti Chhikara, Satender Vinayak, Nishant Pai, Kuldeep Meena, "Improved Apriori Algorithm for Association Rules", International Journal of Technical Research and Applications, Vol. 3, Issue 3, 2015, pp. 238 – 240.

[8] Surajt Chaudhuri, Umeshwar Dayal, "An overview of Data Warehousing and OLAP technology", ACM Sigmond Record, Vol. 26, Issue 1, 1997, pp. 65-74.

[9] Jugendra Dongre, Gend Lai Prajapati, S.V. Tokekar "The role of Apriori Algorithm for finding the association rules in Data Mining", In proceedings of International Conference on Issues and Challenges in Intelligent Computing Techniques, 2014.

[10] Sheila A. Abaya "Association rule mining based on Apriori algorithm in minimizing candidate generation", In proceedings of International Journal of Scientific and Engineering Research, Vol. 3, Issue 7, pp. 171-174, 2012.

[11] S.Appavu, Aravind, Athiappan, Bharathiraja, Muthu Pandian and Dr.R.Rajaram, "Association rule mining for suspicious Email detection:A data mining approach", IEEE Intelligence and Security Informatics, 2007.

Authors' Profiles

Debabrata Datta is presently an Assistant Professor in the Department of Computer Science, St. Xavier's College (Autonomous), Kolkata, India. He has a teaching experience of more than 10 years both at the undergraduate level as well as the postgraduate level of Computer Science and Applications. His research interests include data warehousing and data mining. He has published more than fifteen research papers in different international peer-reviewed journals as well as conferences.

Atindriya De did her B.Sc. with honours in Computer Science from St. Xavier's College, Kolkata, India. She is currently pursuing her Masters in Computer Application under the department of Computer Science and Engineering, Jadavpur University, West Bengal, India.

Deborupa Roy did her B.Sc. with honours in Computer Science from St. Xavier's College, Kolkata, India. She is currently pursuing her Masters in Computer Science from the University of Calcutta, West Bengal, India.

Soumodeep Dutta did his B.Sc. with honours in Computer Science from St. Xavier's College, Kolkata, India. He is currently pursuing his Master's in Computer Science from Institute of Science, Banaras Hindu University, Uttar Pradesh, India.

How to cite this paper: Debabrata Datta, Atindriya De, Deborupa Roy, Soumodeep Dutta,"Apriori Algorithm using Hashing for Frequent Itemsets Mining", International Journal of Education and Management Engineering(IJEME), Vol.8, No.6, pp.46-58, 2018.DOI: 10.5815/ijeme.2018.06.05