iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 4 **Issue:** VII **Month of publication:** July 2016

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

Volume 4 Issue VII, July 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

A Survey on Frequent Pattern Mining Methods

Akansha Pandey[1], Shri Prakash Dwivedi[2], H. L. Mandoria[3]
[1,2,3]Department of Information Technology, College of Technology, GBPUAT

Abstract- In this paper, the frequent pattern mining methods have been discussed that play very important role to generate association rules. The aim of frequent pattern mining is to search repeatedly occurring relationships in a data set. A comparative study between the algorithms from Apriori to more advanced Frequent Pattern growth approach and its variations have been done in this paper. Haoyuan Li et al (2008) suggested that PFP (Parallel Frequent Pattern) algorithm is more efficient as it parallelizes the FP (Frequent Pattern) growth approach. Again, Le Zhou et al (2010) proposed BPFP (Balanced Parallel FP) where the load is balanced to make the algorithm more efficient. The frequent patterns discovered by these algorithms are useful in other data mining applications such as classification, correlations and other relationships among data. The discovery of interesting relationships among a large amount of records can help in decision making process. This paper examines the frequent pattern algorithms, their enhancements, limitations and advantages.
Keywords: Frequent pattern mining, Apriori algorithm, Frequent Pattern growth, PFP, BPFP.

I. INTRODUCTION

A regular and intelligible form or sequence which can find out the difference easily in the manner in which something happens or is done is known as pattern. And those patterns that appear repeatedly in a database are frequent patterns. Taking an example of a set of items, such as oil and spices that appear very often together in a data set is a frequent itemset. And a subsequence, such as buying first a mobile phone, then a memory card, and then its cover, if it occurs regularly in a shopping history, then it is a frequent pattern. Frequent patterns are helpful in mining associations, correlations, data classification, clustering and many other interesting relationships among data. Because of the continuous collection of massive amount of data many industries have become interested in frequent pattern mining techniques. The discovery of interesting relationships among a large amount of records can help in decision making process [7].

Frequent pattern mining concept was first proposed by Agrawal et al. for market basket analysis in 1993. Here, in this method customers' buying habits are analyzed by generating associations among the items placed by customers in their shopping baskets. For example, if customers are buying oil, what is the possibility that they will buy spices also on the same trip to the market. This information helps retailers to develop strategies, by gaining knowledge as to which items are most often bought together by the customers, and increasing sales by doing selective marketing [4].

Taking an example, universe as the set of items available, then for denoting the presence or absence of each item, a Boolean variable is given to that item. Then each basket can be represented by a Boolean vector of values assigned to these variables. The Boolean vectors which are given to each item can be analyzed for buying patterns that represent items that are often purchased together. These often occurring patterns then can be described in the form of association rules.

There are two measures of rule interestingness i.e. rule support and confidence. The support displays the usefulness of found rules and confidence shows the certainty of discovered rules. If the rules satisfy both the thresholds minimum support and minimum confidence which can be set by the users, they become interesting.

Since the first proposal of this frequent pattern mining task and its associated efficient mining algorithms, there have been many publications, on various types of extensions and applications of pattern mining methodologies. Still research is needed to reduce the size of derived pattern sets and enhance the quality of retained patterns.

The main objective of the paper is to understand the basic concepts of frequent pattern mining and the methods used for discovering these frequent patterns. The discovery of such patterns helps to find the association among the itemsets of the query.

The rest of the paper is organized as follows. In section 2, literature review is presented. The section 3 gives apriori algorithm and the work related to it. In section 4, frequent pattern growth algorithm and its extensions is described. Section 5, presents the conclusion.

II. LITERATURE REVIEW

Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules [8]. It is the basic algorithm for finding frequent itemsets which is based on the fact that the algorithm uses prior knowledge of frequent itemset properties [8]. The later variations of Apriori algorithm, partitioning technique was proposed by Savasere,

Omiecinski, and Navathe in 1995 [1] to improve the algorithm's efficiency. This new proposed algorithm reduced the I/O overhead as well as the CPU overhead for most cases. The sampling approach is discussed by Toivonen in 1996 [3] for large databases. The approach here is to pick a sample randomly and then finding all the association rules that probably present in the whole database based on that sample and verifying the results with the rest of the database. A dynamic itemset counting approach is given by Brin, Motwani, Ullman, and Tsur in 1997 [10]. The idea behind dynamic itemset algorithm is to find large itemsets with minimum passes than the traditional algorithms and yet using fewer candidate itemsets than the methods based on sampling. Also the concept of item reordering was introduced in this paper which improves the low-level efficiency of the algorithm.

Many frequent itemset mining methods have been proposed as alternatives to the Apriori-based approach. A pattern-growth approach i.e. FP growth was proposed by Han, Pei, and Yin in 2000 [5] for mining frequent itemsets without candidate generation. This frequent pattern tree structure is an extended prefix tree structure which is used for storing compressed, crucial information about frequent patterns, and develop an efficient FP-tree based mining method, FP-growth, for mining the complete set of frequent patterns by pattern fragment growth. Later to improve its efficiency the algorithm is parallelized on distributed machine. The algorithm PFP (Parallel Frequent Pattern) growth is proposed in 2008 by Haoyuan Li et al. [2], where the partitioning applied eliminates computational dependencies between machines and hence the communication between them. Again the parallelized algorithm is improved by balancing the load and the algorithm is proposed as BPFP (Balanced Parallel FP) in 2010 by Le Zhou et al. [6]. Because of the load balancing feature this improves parallelization and hence improves performance.

III. APRIORI ALGORITHM

Apriori algorithm was first proposed by R. Agrawal and R. Srikant for mining frequent itemsets for Boolean association rules in 1994 [8]. It is based on the concept that previous information of frequent itemset properties is used. It is an iterative approach, where to find (m+1) itemsets, m-itemsets are used. First step is to find the set of frequent itemsets by scanning the database to have the count for each item, and those items that have count more than minimum support threshold are selected, that is indicated by L_1. Next, this L_1 is used to find L_2, which is then used to find L_3 and this process goes on, until no more frequent m-itemsets can be found. The finding of each L_m requires one full scan of the database.

For reducing the space, Apriori property is used that states all the subsets of a frequent itemset that are nonempty must also be frequent.

Apriori algorithm is a two-step process join and prune.

A. Join

To find a set of candidate m-itemsets(L_m) is created by joining L_{m-1} with itself, represented by C_m. Let l_1 and l_2 be itemsets of L_{m-1}. $l_i[j]$ denotes to the jth item in L_i. It is assumed in this algorithm that the items within an itemset are sorted in lexicographic order. The join, L_{m-1} join L_{m-1}, is executed, where members are joinable if first (m-2) items in both the sets are same.

B. Prune

C_m is a superset of L_m. A database scan is done to determine the count of each candidate in C_m would result in the determination of L_m. C_m can be huge, and so this could involve heavy computation. To reduce the size of C_m, the Apriori property i.e. any (m-1)-itemset that is not frequent cannot be a subset of a frequent m-itemset.

1) Variations of Apriori Algorithm

a) *Hash-based technique*: This technique is used to reduce the size of the candidate m-itemsets, C_m, for m > 1. For example, while scanning each transaction in the database to generate the frequent1-itemsets, all the 2-itemsets can be created for each transaction, then mapped into the different buckets of a hash table structure, and the bucket counts is increased accordingly. A 2-itemset with a bucket count that does not satisfy the support threshold cannot be frequent and removed from the candidate set [9]. Hence, the number of candidate itemsets can be decreased.

b) *Transaction reduction:* A transaction that does not contain any frequent m-itemsets cannot contain any frequent (m+1)-itemsets according to apriori property. Hence, such a transaction can be removed from further consideration because subsequent database scans for j-itemsets will not need to consider such a transaction.

c) *Partitioning:* It involves only two database scans to mine the frequent itemsets [1]. It consists of two phases, in the first phase the transactions are divided into m non-overlapping partitions. If the minimum support threshold is represented by min sup, then for a partition the minimum support count is min sup * the number of transactions in that partition. Any itemset that is

possibly frequent in the database then that must occur in at least one of the partitions as a frequent itemset. In the second phase, a second scan of database is conducted in which the actual support of each candidate is assessed to determine the global frequent item sets.

d) *Sampling:* Here, in this approach for searching frequent itemsets instead of the whole database the concept is to pick a random sample of the given data and then search is performed in that sample [3]. Because the searching for frequent itemsets is performed in a sample, it is possible that some of the frequent itemsets are missed. To reduce this problem the support threshold lower than minimum support is used to find the frequent itemsets in the sample. The remaining database is then used to compute the actual frequencies of each itemset in the frequent itemset evaluated for the sample.

e) *Dynamic itemset counting:* Here the database is divided into several blocks which are marked by the start points [10]. The main concept is to add new candidate itemsets at any start point which checks new candidate itemsets only immediately before each complete scan of the database. The lower bound of the actual count for this technique set as the count-so-far. If the count-so-far of the itemset exceeds the minimum support threshold, then the itemset is considered the frequent itemset.

2) *Limitations of Apriori Algorithm*

a) After this much improvement in the algorithm it still generates a large number of candidate sets.

b) And the need of scanning whole database and checking of large set of candidates repeatedly is not reduced.

IV. FREQUENT PATTERN GROWTH

This approach is based on divide-and-conquer strategy. The first step is to compress the whole database into a frequent pattern tree that preserves the association information of itemsets. The next step is to divide this compressed database into a set of conditional databases, where each conditional database is associated with one frequent item and also these databases are mined separately. Because for each frequent item its associated data sets are needed to be examined only. This approach is beneficial as it reduces the size of the data sets to be searched [5].

One of the advantages of FP-growth method is that the problem of finding long frequent patterns is changed into searching for shorter ones because of small conditional databases repeatedly and then concatenating the suffix. For good selection and less search costs, the least frequent items are used as a suffix.

When the database is large, the alternative of memory based FP tree is to first divide the database into a set of small databases, and then an FP-tree is constructed and mined in these small databases separately. This process can be applied repeatedly to any of the small database if its FP-tree is still not fit in the main memory.

A. *Variations of FP growth*

1) *Parallel FP growth:* Parallel FP growth was introduced to overcome the limitations of FP growth. In FP growth for a large scale database, support threshold need to be set large enough, or the FP-tree would overflow the storage. For Web mining tasks, the support threshold is set to be very low to obtain long-tail itemsets. This setting may require unacceptable computational time. While in parallel FP growth approach, all the steps of FP growth can be parallelized leading to decrease in computational time [2].The PFP approach has five steps:

a) *Sharding:* The database is divided into successive parts and stored on different computers. This distribution and division of data is called sharding.

b) *Parallel Counting:* This step is to count the support values of all the items that appear in database. The input is one shard of database. In this step, the result of discovered items is stored in F-list.

c) *Grouping Items:* Here, in this step all the items on F-List are divided into Q groups, where a unique group id (gid) is given to each group. As both the lists are small and the time complexity is linear, hence this step does not take much time for computation.

d) *Parallel FP-Growth:* This step performs the following two functions:

i) *Generating group-dependent transactions:* The input is given a shard of database which is generated in the sharding step. Before any transactions in the shard, it first reads the G-list. And then, it outputs key-value pairs, where the key is a group-id and the value is a group-dependent transaction.

ii) *FP-Growth on group-dependent shards:* When all group dependent transactions are generated, for each group-id, then this step is performed where a local FP-tree is built and growth its conditional FP-trees recursively for each shards. During this recursive

process, the discovered patterns can be generated as output.

iii) Aggregating: In this step, the output generated in the previous step is aggregated and final results are generated.

2) *Balanced Parallel FP growth:* There are two differences as compared with PFP. First is that the grouping step has a balanced strategy, where the entire mining task is divided into even subtasks to improve the parallelization [6]. And second is that BPFP doesn't involve the Aggregating step [6], as its aim is to discover all the frequent itemsets.

The BPFP approach involves the following steps:

a) *Sharding:* The whole database is first divided into successive parts and stored on different computers.

b) *Parallel Counting:* The input of this step is one shard of the database and the output is the F-list which contains the list of frequent items sorted in descending order, with respect to the frequency.

c) Balanced Grouping: This step is different from PFP as, all the items of F-list are fairly divided into Q groups by balancing the load among the groups. This step is divided into following two steps:

i) *Mining Load Estimation:* For computing load of each item some estimation is needed. First, load of FP-Growth on conditional pattern base is estimated by number of recursive iterations during FP-Growth execution on conditional pattern base of each item. The second is to estimate the location of each item in F-List by the length of the longest frequent path in the conditional pattern base. The estimated load of item i (T_i), can be computed by location of item i in F-List (L_i), as follows:

$$T_i = \log L_i$$

ii) *Balanced Partition:* By the load computed in previous step all the items are sorted in descending order forming L-List. Then, the items are arranged in one item per group as the front-most Q items from the initial Q groups. Load of each group is initialized with load of the item it contains. Then the following steps are repeated until all items in the L-List are grouped:

The next non-grouped item is added to the group with the minimum load in L-List.
The load of that group is increased by load of the new item.
Parallel FP Growth: The following two steps are performed:
Generating group-dependent transactions: The following two operations are performed for each T_i:
For each group which contains item in Ti, locate its right-most item in Ti, say L.
Output one key-value pair <key' = group-id, value = {Ti[1], Ti[2], ... , Ti[L-1], Ti[L]}>.

FP-Growth on group-dependent transactions: In this step, the input is provided in the form of <key' = group-id, value' = DB (group-id)> one by one, where each DB (group-id) are all transactions of the same group. For each DB (group-id), the FP-Tree is constructed and mined recursively. The only difference from traditional FP-Growth is that only items in group group-id are traversed in the first FP-Growth execution.

V. CONCLUSION

In this article, a brief overview of the frequent pattern mining methods that are apriori and frequent pattern growth and their variations is presented. To compress the original transaction database Frequent Pattern growth approach constructs a highly compact data structure, rather than employing the generate-and-test strategy of Apriori-like methods. It is analysed that the frequent pattern growth approach is more efficient than apriori. And the parallelized FP growth takes less computational time as compared to the traditional approach. The new balanced parallel approach further improves the performance by balancing the load of the parallel FP growth. After reviewing the papers on frequent pattern mining it is felt that the bottleneck of frequent pattern mining depends on whether compact and high quality set of patterns can be derived that are most useful in applications. Still research is needed to reduce the size of derived pattern sets and enhance the quality of retained patterns.

S. no.	Algorithm	Features	Advantages	Limitations
1	Apriori	i. Based on prior knowledge of data. ii. Uses bottom-up generation approach for frequent itemsets combinations.	i. The main advantage is that k items candidate itemsets can be generated by joining large itemsets having k-1 items, and removing those	i. It is very much costly to handle a large number of candidate sets. For example, if there are 10^4 frequent 1-itemsets, the Apriori algorithm

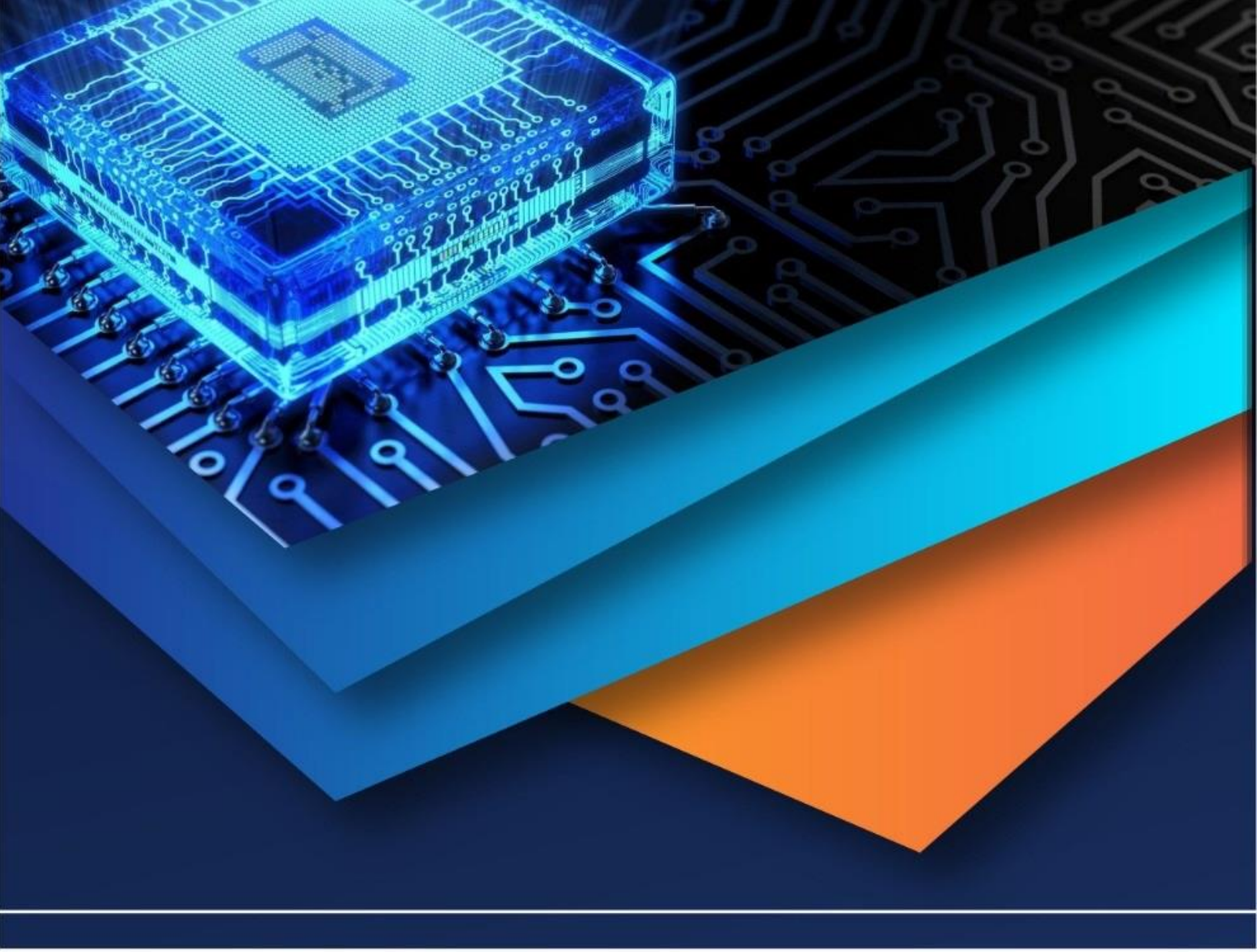
		iii. Assumption taken in this algorithm is that the itemsets are stored in lexicographic order. iv. All nonempty subsets of a frequent itemset must also be frequent.	that contain any subset that is not large which implies that the itemsets are not frequent in that subset. This results in the generation of less number of candidate sets.	will need to generate more than 10^6 length-2 candidates and accumulate and test their occurrence frequencies. ii. It is tedious to scan the database recursively and check a huge set of candidates by pattern matching.
2	Hash based technique	i. The itemsets are hashed in the corresponding buckets.	i. The number of itemsets decreased as compared to the Apriori algorithm. ii. Less execution time in comparison with Apriori.	i. Slightly higher cost in the first iteration due to the generation of hash table.
3	Transaction reduction	i. Depends on the transactions instead of the itemsets.	i. Reduces the number of transactions to be scanned in the future iterations.	i. Does not consider the itemsets as the priority.
4	Partitioning	i. Partitions database to find candidate itemsets which are the local frequent itemsets in the partitions. ii. The local frequent itemsets then combined to form candidate itemsets and getting the output of global itemsets.	i. Reduction in I/O overhead as well as in CPU overhead in comparison with previous algorithms.	i. Partition size and the number of partitions can't be taken large. The size to be considered so that each partition can fit into main memory and therefore be read only once in each phase.
5	Sampling	i. Mining on a random set of data in place of the given data.	i. Beneficial when the efficiency is of much importance.	i. Possibility is that some of the global frequent itemsets can be missed. ii. Accuracy might not be met due to the consideration of sample instead of the whole database.
6	Dynamic itemset Counting	i. Candidate itemsets can be added at different points during a scan.	i. Fewer database scans when compared to the traditional approaches for finding all the frequent itemsets. ii. Item reordering concept is added that improves the low level efficiency of the algorithm.	i. Candidate set generation is still costly, especially when there exists long patterns.
7	Frequent Pattern	i. Tree structure for storing	i. Avoids costly, repeated	i. When the database is large, it

	Growth	compressed, crucial information about frequent patterns.	database scans. ii. Avoids the costly generation of a large number of candidate sets.	is sometimes unrealistic to construct a main memory based FP-tree.
8	Parallel Frequent Pattern	i. Parallelizes the FP-Growth algorithm on distributed machines.	i. Partitioning applied in this algorithm eliminates computational dependencies between machines and hence the communication between them. ii. Computational time is linear.	i. PFP doesn't take into consideration load balance which is quite important for large scale data processing.
9	Balanced Parallel Frequent Pattern	i. BPFP adds into PFP load balance feature.	i. Because of the load balancing feature this improves parallelization and hence improves performance.	i. The precision for balanced grouping strategy is not taken into consideration.

Table 1. Comparison between the frequent itemset mining algorithms

REFERENCES

[1] Savasere, E. Omiecinski, and S. Navathe (1995). "An efficient algorithm for mining association rules in large databases". In Proc. 1995 Int. Conf. Very Large Data Bases, pp. 432–443.
[2] Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang, Edward Chang (2008). "PFP: Parallel FP-Growth for Query Recommendation". In Proc. of the 7th Pacific-Asia conference on Advances in knowledge discovery and data mining, IEEE, pp. 467-473.
[3] H. Toivonen (1996). "Sampling large databases for association rules". In Proc. 1996 Int. Conf. Very Large Data Bases, pp. 134–145.
[4] Jiawei Han, Hong Cheng, Dong Xin, Xifeng Yan (2007). "Frequent pattern mining: current status and future directions". Data Mining and Knowledge Discovery, pp. 55-86.
[5] J. Han, J. Pei, and Y. Yin (2000). "Mining frequent patterns without candidate generation". In Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data, pp. 1–12.
[6] Le Zhou, Zhiyong Zhong, Jin Chang, Junjie Li, Huang, J.Z., Shengzhong Feng (2010). "Balanced parallel FP-Growth with MapReduce". Information Computing and Telecommunications, 2010 IEEE Youth Conference, pp. 243-246.
[7] Ramya. S. Bhat, A. Rafega Beham (2016). "Comparative Study on Algorithms of Frequent Itemset Mining". International Journal of Computer Science and Mobile Computing, 5, pp. 271-275.
[8] R. Agrawal and R. Srikant (1994). "Fast algorithms for mining association rules". In Proc. 1994 Int. Conf. Very Large Data Bases, pp. 487–499.
[9] R. Agrawal, M. Mehta, J. Shafer, R. Srikant, A. Arning, and T. Bollinger (1996). "The Quest data mining system". In Proc. 1996 Int. Conf. Data Mining and Knowledge Discovery, pp. 244–249.
[10] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur(1997). "Dynamic itemset counting and implication rules for market basket analysis". In Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data, pp. 255–264.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY