Journal of Theoretical and Applied Information Technology
15th February 2018. Vol.96. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

FREQUENT ITEMSET MINING ALGORITHMS: A SURVEY

[1] Sireesha Moturi, [2] Dr.S.N.TirumalaRao, [3] Dr. Srikanth Vemuru

[1]Research Scholar, K L E F. Astt. Prof., Department of Computer Science and Engineering, NEC

[2] Prof., Department of Computer Science and Engineering, Narasaraopeta Engineering College

[3] Prof., Department of Computer Science and Engineering, K L E F

E-mail: [1]sireeshamoturi@gmail.com, [2]naga_tirumalarao@yahoo.co.in, [3]vsrikanth@kluniversity.in

ABSTRACT

Task of extracting fruitful knowledge from huge datasets is called data mining. It has several aspects like predictive modeling or classification, cluster analysis, association analysis, anomaly detection and regression etc. Among all association rule mining is one of the major tasks for data mining. Association analysis is mainly used to discover patterns, which describes strongly associated features in the data. Market basket data is one of the major applications of association rule mining. Other applications include bioinformatics, medical diagnosis, scientific data analysis, web mining, finding the relationships between different elements of earth climate system etc. Various algorithms have been proposed by researchers to improve the performance of frequent pattern mining such as Apriori, Frequent Pattern (FP)-growth etc. We are providing a brief description of some of the techniques in detail in the later section of this paper.

Keywords: *Association Rule Mining, Support, Confidence, Frequent Itemset*

1. INTRODUCTION

In today's competitive world, every business organization has immense competition to attain success. So every business organization or business executive makes strategic decisions in order to achieve success. In present scenario, a vast amount of data is generated by daily activities such as Science, Engg., Business and many other areas, due to the rapid advances in computerization and digitalization techniques. Sometimes, users might have no idea about what kind of patterns in their data may be more useful, and thus might prefer to discover various kinds of patterns in parallel. Therefore, mining the relevant and important information from this huge amount of data is necessary for many industries which can help in decision making process [2]. So, the most challenging task is to extracting useful knowledge from that vast amount of data. Thus, it is essential to have a data mining system which will mine various kinds of patterns to fulfill the needs of different users or applications. So, the data mining is a process which uses wide variety of tools to extract knowledge from massive datasets.

Data mining plays a crucial role in the knowledge discovery process by finding hidden patterns, associations, constructing analytical models, performing classification and prediction, performing clustering and presenting the mining results by using visualization tools and techniques. So the data mining system should be able to discover patterns at various levels of granularities to accommodate different user expectations or applications. Data mining is also called as Knowledge Discovery in Databases (KDD) why because it integrates different techniques form different disciplines such as neural networks, statistics, machine learning, database technology and information retrieval, etc.[1]. Data mining is extensively used in banking and financial services, consumer goods and retail distribution sectors and controlled manufacturing etc.

2. ASSOCIATION RULE MINING

Association rule mining is one of the major techniques in data mining. Association rule mining is used to find associations, frequent patterns, correlations and/or informal structures from transactional databases or other information repositories[3].It is useful in various areas like retail stores, inventory control, marketing, catalog design, selection of crops in agriculture sector, wireless sensor networks, bioinformatics, telecommunication alarm diagnosis, web mining

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

and scientific data analysis like earth science data etc. The main task of association rule mining is to search for interesting relationships among sets of items in a given database.

Various algorithms were proposed for finding the frequent itemsets. Those algorithms are divided into two groups. One is the Apriori-like approach which follows horizontal mining technique and another one is the FP-growth-like approach which follows vertical mining technique. In the Apriori-like approach candidate patterns of length (k+1) are generated by using the frequent patterns of length k and it follows breadth-first search strategy. However to generate the longest frequent itemset, it requires multiple scans of database. Some studies, such as [3] [5] [18] [22] [30] [26] [29] adopt the Apriori-like approach. To improve the performance of Apriori, the FP-growth-like approach is introduced by using a data structure FP-tree (Frequent Pattern tree) to store the database and it follows depth-first search strategy to find frequent patterns. In the FP-growth methodto find the frequent patterns, only two scans of database are enough and there is no candidate itemset generation. So, FP-growth is much faster than Apriori. There are several alternatives and extensions to FP-growth, such as [9] [11] [14][15][16][20] [21] [24] [25] [27][28].

Generally, association rule mining can be defined as a two-step process.

Step1: Find all frequent itemsets.

Step2: Generate all association rules.

The overall performance of association rule mining is determined by first step. After finding the frequent itemsets[2], the association rules are generated[5].

3. FREQUENT ITEMSET MINING

The frequent itemset mining can be formally stated as follows. Transactional database (TB) contains set of transactions with unique ID's called Transaction Identifier (TID). Each transaction contains collection of zero or more items called as an itemset. An itemset with zero elements is called null itemset. If an itemset with 'n' elements is called as an 'n'-itemset. Every subset of I is an itemset.

Let us consider I={I1,I2,....,Im}be the set of items and TB={T1,T2,...,Tn}be the task relevant data, where each transaction Ti is a subset of items chosen from I, such that Ti€I. Let X be an itemset and a transaction T is said to contain X if and only if X I. The support count of an itemset X in TB is the number of transactions in TB, X occurs as a subset. For a given TB, let minsup be the minimum support threshold value specified by the user. If support_count(X) >= minsup, then itemset X is called a frequent itemset. The support of an itemset is represented with σ. σ(X) of an itemset X is defined as follows.

$$\sigma(X) = ||T \in TB|\, X \subseteq T|\,/\,|D|$$

Given a transactional database with minimum support threshold, the task of frequent pattern mining is to find all frequent patterns from the transactional database. Consider the transactional database shown in Table-1with minimum support threshold=2.

Table-1: Transactional Database

TID	ITEMS
T_1	ABCEFH
T_2	ACG
T_3	E
T_4	ACDEG
T_5	ACEG
T_6	E
T_7	ABCEF
T_8	ACD
T_9	ACEG
T_{10}	ACEG

Table-2 shows the set of all frequent patterns discovered from the transactional database shown in Table-1

Table-2: Frequent patterns of Table-1

All frequent Patterns with minsup=2
A: 8, B:2, C:8, D:2, E:8, F:2, G:5
AB:2, AC:8, AD:2, AE:6, AF:2, AG:5, BC:2, BE:2, BF:2, CD:2, CE:6, CF:2, CG:5, EF:2, EG:4
ABC:2, ABE:2, ABF:2, ACD:2, ACE:6, ACF:2, ACG:5, AEF:2, AEG:4, BCE:2, BCF:2, BEF:2, CEF:2, CEG:4
ABCE:2, ABCF:2, ABEF:2, ACEF:2, ACEG:4, BCEF:2
ABCEF:2

4. ASSOCIATION RULES

Association rule is an implication expression and display it in the form of a rule X⇁Y, where X and Yare disjoint subsets and X I and Y I where I = {I1, I2, ..., Im} i.e X∩Y=Φ. For a given dataset TB, the minconf be the minimum confidence threshold value specified by the user. If confidence(X⇁Y) >=minconf then X⇁Y is called strong association

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

rule. The confidence of a rule X\Y is calculated as follows.

$$C(X\backslash Y) = \sigma(XUY)/\sigma(X) \text{ or } P(Y/X)$$

The strength of an association rule can be measured in terms of support and confidence. These measures are explained in detail in [4].The rule X\Y holds on the transactional database TB with support S and confidence C, where S is the percentage of transactions that contain XUY, and C is the percentage of transactions in TB containing X that also contains Y. The rules which satisfy both support and confidence thresholds are called strong association rules.

5. LITERATURE SURVEY

One of the first algorithms for association rule mining was AgrawalImielinski Swami (AIS) algorithm [2]. In AIS algorithm the candidate itemsets are generated and counted on-the-fly basis when the database is scanned. New candidate itemsets are generated by extending the large itemsets with other items in the transaction. The drawback of AIS is it makes multiple passes over the database that results in unnecessarily generating and counting too many candidates that turn out to be small.

Like AIS, in Set-oriented Mining (SETM) also the candidate itemsets are generated on-the-fly basis when the database is scanned, but counted at the end of the pass. Standard Structured Query Language (SQL) join operator is used for candidate generation. When the New candidate itemsets are generated, the TID of the generating transaction is saved with the candidate itemset in a sequential structure. At the end of the pass, the support count of candidate itemsets is obtained by aggregating this sequential structure. SETM also have the same drawback of AIS. In addition to this for each candidate itemset, there are many entries as its support value.

One of best known algorithm for finding the frequent itemsets is the Apriori algorithm[5]. Apriori is a Breadth First Search algorithm which uses level wise search to find out the frequent itemsets i.e frequent-k itemsets are used to generate frequent k+1 itemsets This is based on the candidate-generation and test approach i.e first generate candidate 1-itemsets by scanning the transactions T of database TB and placed in C1. The itemsets which satisfy the minimum support threshold are called frequent-1-itemsets, placed in level wise relation L1. The frequent-1 itemsets in L1are used to generate frequent-2-itemsets. This process is repeated until there are no more frequent itemsets. Apriori algorithm works in two step process named as join step and prune step[6].

1. Join step: Self join of Lk-1 with itself is performed to form Lk. From candidate k-itemsets i.e the candidates l1 and l2 of Lk-1 are joined if and only if (l1[1]=l2[1]) (l1[2]=l2[2]) … (l1[k-2]=l2[k-2]) (l1[k-1]=l2[k-1]). Then the resulting itemset l1[1], l1[2], …, l1[k-2], l1[k-1], l2[k-1] is formed by joining l1 and l2.
2. Prune step: This step is mainly used to eliminate some of the candidate kitemsets which are infrequent i.e CK is the superset of LK, but its members may or may not be frequent. After this step all frequent k-itemsets are placed in LK.

In this algorithm, to find frequents patterns downward closure property is used, i.e all subsets of a frequent itemset must themselves be frequent or all supersets of infrequent itemset must be infrequent.

The major flaw of this algorithm is multiple database scans, i.e if the database is too large, it takes more time to scan the database in order to calculate the support count of every candidate. To overcome this problem several variations of Apriori are developed.

R.Agarwal, R.Srikant proposed AprioriTid[5] algorithm which uses the Apriori candidate generation function. The main feature of this algorithm is the transactional database TB is not used for support counting after the first pass. Instead it uses CK' .The entries in CK' is of the form, where XK is the largest k-itemset. If a transaction does not have any candidate k-itemset, then there is no entry in CK' for that transaction. As the value of k increases the number of entries in CK' are decreased. However, number of entries in CK' may be more for smaller value of k. It means Apriori outperforms AprioriTid for initial passes i.e for smaller value of k, but for larger value of k AprioriTid outperforms Apriori.

Another algorithm known as Apriori hybrid was also presented by R.Agarwal, R.Srikant[5]. Apriori hybrid is a combination of Apriori and AprioriTid. It uses Apriori, in the earlier passes and later it switches to AprioriTid when it expects that the CK' is fit in main memory.

These three algorithms Apriori, AprioriTid and Apriori hybrid outperforms the previous known frequent itemset mining algorithms named as AIS and SETM.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Improved Apriori Algorithm (IAA) [18] based on the original Apriori algorithm is introduced by Huan Wu et al. by using a new count based method in order to prune the candidate itemsets and uses new record generation method in order to reduce the size of the data scan. By using these two methods IAA outperforms other algorithms like Apriori, AprioriTid, HDO[19] etc.

o reduce the scanning time and to reduce the redundant generation of candidate itemsets an improved Apriori algorithm called Transaction Reduction [23] is introduced by Jaishree Singh, Hari Ram, and Dr.J.S.Sodhi. In this method an attribute Size OfTransaction (SOT) is introduced. According to the value of k, the transactional database is updated by deletion process. According to the value of k, algorithm searches for same value of SOT in database. If it matches then delete only those transactions from database, then that leads to reduced scan time and less number of candidate itemsets. So, it reduces the total I/O time required. However it has the overhead to manage the new database after updating the database every time. So, there should be some approach which has very less number of database scans.

The classical Apriori algorithm is inefficient, because it takes so much time to scan the database, and is inefficient due to several scans of database. To overcome these limitations, a new algorithm Transaction Reduction- Bit Array Matrix (TR-BAM)[26], [31] is developed by Vijayalakshmi et al. The entire database is scanned only once and then the data is represented in the form of a Bit Array Matrix. The transactions which are repeated in the database are represented by the Repetition Count (RC) column and a new row sum is used to store the number of nonzero elements in the column. The support count of k-itemset can be obtained by performing bitwise AND operation on the bits of corresponding items and by using the count value in the RC column. By using these two techniques RC and sum. TR-BAM method greatly reduces the amount of time and space required. The time consumed between original Apriori and TR-BAM is greatly reduced when the value of support increases.

To avoid the costly candidate itemset generation and test process completely two new algorithms called CountTableFI (CTFI) and Binary CountTableFI (BCTFI) [30] are recommended by Marghny H. These algorithms represent the transactional database in the form of a binary number and decimal number. In CTFI the original transactional data is transformed into smaller transactional data along with the information of frequent itemsets. CTFI algorithm is based on the set and subset properties. If multiple transactions having the same set of items merge those transactions and represent the number of occurrences as count. If multiple transactions having the common prefix and one transaction is subset of another according to some sorted order, then merge the shared part as long as the count of items is registered properly. Here the Bitwise AND operation is used for intersection operation for quick generation of frequent itemsets. In BCTFI, first the original transactional database is transformed to binary data. By using this binary data the original data is transformed to decimal number. The BCTFI is similar to CTFI, but the merge process in BCTFI is based on the decimal value of each transaction. So, it is easy to identify the identical transactions by using the decimal number. These two algorithms does not need to generate candidate itemsets, so it reduces the costly database scans. Experimental results shown that CTFI, and BCTFI outperforms most of the Apriori-like algorithms and FP-growth algorithm in most of the cases.

Improved Apriori algorithm named BE-Apriori [29] based on pruning optimization and transaction reduction is introduced by Zhuang Chen, et al. By using this improved algorithm, the number of frequent itemsets becomes much less, therefore a significant reduction in running time. Pruning optimization strategy states that to generate candidate-K itemsets on the basis of candidate-(K-1) frequent itemsets, a temporary table is used to count the frequency of all the items in frequent itemsets. This method reduces the number of frequent itemsets generated. Transaction reduction strategy is used to reduce the number of transactions to be scanned by compressing the transactional database. In this way, BE-Apriori algorithm improves the efficiency of Apriori by reducing the number of candidates generated and a significant improvement in running time by reducing the transaction length.

To overcome the problem found in many Apriori-like algorithms i.e candidate set generation and test approach, a new algorithm to find frequent patterns of length K without candidate generation is Frequent Pattern growth (FP-growth) introduced by the Jiawei Han et al.[7] based on Depth First Search method. The FP-growth algorithm compresses the dataset using a compact data structure called as Frequent Pattern tree (FP-tree) and directly extracts frequent patterns from an FP-tree by exploring the tree in a bottom-up fashion, which avoids costly repeated database scans. To avoid the generation of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

a large number of candidate itemsets it uses frequent pattern growth method. To reduce the search space it adopts a divide-and-conquer strategy to decompose the mining task into a set of smaller tasks. In most of the cases, FP-growth outperforms the Apriori-like algorithms by several orders of magnitude. This method saves considerable amount of memory for storing the transactions, therefore it substantially reduces the search costs. Finally, it reduces both the time and space complexity required when compared to Apriori. However, if the database is too large, it is unrealistic to construct a main memory based FP-tree.

A novel vertical data mining algorithm called Diffset[8] using vertical data format was proposed, which only maintains the differences in the TId's of a candidate patterns from its generating frequent patterns i.e it avoids storing the entire TIDset for every member of a class. So, this method drastically reduces the size of memory required by orders of magnitude to store intermediate results that significantly increase the performance.

MingjunSong, and Sanguthevar Rajasekaran[9] recommended a novel algorithm called Transaction Mapping Algorithm by exploring the vertical data representation for Frequent Itemsets Mining. The transaction tree is used to represent all the transactions in the database. Each node in the tree has the name of the item and count that keeps track of number of transactions that contain this item. After constructing the transaction tree the transactions ids of each itemset are mapped and compressed to transaction intervals. The counting of these itemsets is performed by intersecting these interval lists in a depth-first manner along with the lexicographic tree. This compression technique greatly reduces the intersection time.

M. J. Zaki [10] worked with two new algorithms Eclat and MaxElat for fast association rule mining. In these algorithms first itemsets are clustered by using equivalence classes and then frequent itemsets are extracted from each cluster by using bottom-up or hybrid traversal. To enhance the performance of association rule mining a new data structure named as Support-Ordered Trie Itemset (SOTrieIT) is introduced by Yew-KwongWoon et al. [14] with two algorithms named as Fast Online Dynamic growth (FOLD-growth) and Fast Online Dynamic Association Rule Mining-2 (FOLDARM2). First SOTrieIT is constructed by extracting 1-itemsets and 2-itemsets. SOTrieIT is ordered by decreasing support count and every node of SOTrieIT is represented with an item name and its support count. SOTrieIT contains only two levels of nodes. First level node of 8 SOTrieIT is the support count of 1-itemset and second level node of SOTrieIT is the support count of 2-itemset. FP-growth is good in situations where kmax >10, while FOLDARM is extremely fast when the kmax of largest frequent itemset is small and thus, FOLD-growth is an attempt to amalgamate both (FP-growth and FOLDARM) algorithms strengths. In FOLD-growth L1 and L2 are found quickly with the SOTrieIT and they can be used to prune the transactions which are used to construct the FP-tree. Hence, only one scan of database is enough to build the FP-tree. If L2 is not found, immediately terminate the algorithm because all possible frequent itemsets in L1 are already obtained. When the transaction are added or deleted, the SOTrieIT can be incrementally updated, but FP-tree can be reconstructed always whenever the database is updated. In this way FOLD-growth achieves better performance than Apriori and FP-growth. FOLDARM uses the SOTrieIT for quick generation of L1 and L2 like FOLD-growth. To find the remaining itemsets it uses Apriori. FOLDARM2 is similar to FOLDARM but it uses an additional step to reduce the size of the database. FOLD-growth is nearly 100 times faster than FP-growth with the SOTrieIT data structure. The main strength of SOTrieIT is speed in discovering L1 and L2, while the main weakness is it discovers only L1 and L2. FOLD-growth uses SOTrieIT, so it can be more incremental than FP-growth to certain extend.

JieDong and Min Han presented an effective algorithm named BitTableFI[16] to mine frequent itemsets, to compress the database and for quick candidate itemset support count generation a special data structure named BitTable is used horizontally and vertically. First generate the frequent 1-itemsets and then initialize the BitTable database. If an item 'i' appears in transaction 't', the bit 'i' of BitTable's TB element is set to one otherwise set to zero[17]. For every frequent 1-itemsets, the binary values of the item are converted to an integer value and that is added to the BitTable database. In this manner the entire database is compressed to a BitTable database, so that it can be fit in main memory. The candidate itemset support count is calculated quickly by accessing the support of the candidate itemsets directly from the BitTable database and performing bitwise AND operation on corresponding elements of the each candidate itemset. Then that candidate itemset is said to be frequent if it satisfies the minimum support threshold.

The major advantages of this method are

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

1) It construct special BitTable database which is much smaller in size when compared to the original database.
2) It uses bitwise AND operation to generate the frequent candidate itemsets.

This Bitwise AND operation is much faster than the traditional candidate itemset generation mechanisms. However, the BitTable is only focusing on candidate itemset generation and support counting issues, but it does not concentrate on any other techniques like to reduce the size of the candidate itemset and number of times scanning the database etc.

A novel vertical mining algorithm called PrePostVertical (PPV) [20] is presented by Zhihong Deng et al. for fast frequent itemset discovery. It uses a new data structure Node-list, which is obtained from PrePostCode-tree (PPC-tree). PPC-tree is constructed to store the database and traversed in preorder and postorder manner. Every node in a PPCtree is assigned with preorder code, postorder code, count, item name, and children node list. The PP-code of any node N in the PPC-tree is represented as <(N.Preorder,N.Postorder):N.count>. Every frequent 1-itemset can be represented as a Node-list by using the PP-codes and count from the PPC-tree. The Node-list of a frequent item is the sequence of all the PP-codes for that item from the PPC-tree. Generally, the Node-list is represented as below.

$\{ (a_1,b_1):c_1>,<(a_2,b_2):c_2>,\ldots<(a_n,b_n):c_n> \}$

Where 'a' is its preorder value, 'b' is its postoder value and 'c' is its support count. The support of an item 'i' is obtained by adding c1+c2+….+ cn. To generate frequent patterns of length (k+1) it performs intersection operation on Node-lists of frequent patterns of length k. The Node-list is more efficient because transactions with common prefixes share the same nodes of the PPC-tree and support count is calculated by performing intersection on the Node-lists. The ancestor-descendent relationship of two nodes can be effectively verified by prepost codes of nodes in the PPC-tree. The experimental results show that PPV outperforms other vertical mining algorithms like FPgrowth, Eclat and dEclat.

For mining frequent itemsets, a novel algorithm named BitApriori[21] is invented, by describing the transactional database in the form of a binary string. First frequent-1, and frequent-2 itemsets are generated, then binary string is used to record each of the frequent-2 itemsets. The trie is constructed with the binary string in each leaf node, and then the trie is extended by using a special technique named as equal support pruning. The K^{th} layer of the trie is generated by combining each node 'm' in the $(K-1)^{th}$ layer of the trie with one of its sibling nodes 'n'. Then check all of its (K-1) subsets. If one of the (K-1) subsets is infrequent, then all of its supersets are infrequent. If any of its (K-1) subsets is in the equal support set, the item in node 'n' is placed into the equal support set of p. If none of the (K-1) subsets are in the equal support set or infrequent the logical AND operation is performed between the binary strings of node 'm' and 'n'. If the support is larger than minimum support, or not equal to the support of 'm', then a new child node is created for 'm' with the item in node 'n' and inserted into the trie. BitApriori outperforms Apriori because it just scans the database twice for candidate generation and traverse the trie only once for support counting. However, when the database is large BitApriori suffers with the problem of memory scarcity and when the trie has many nodes from the root node, BitApriori is not effective.

BitApriori algorithm[21] was modified by Zubair Khanet al. to find the frequent patterns, named Modified BitApriori[25] in order to improve the efficiency based on the trie data structure. The major difference between BitApriori and Modified BitApriori is that Bitwise AND operation on binary strings is performed in BitApriori[21], Bitwise OR operation on binary strings is performed in Modified BitApriori. When the minimum support threshold is low, the Modified BitApriori outperforms the Apriori.

To reduce the cost of candidate generation and test [22] a new algorithm IndexBitTableFI [23] is presented by Wei Song et al. It uses the BitTable horizontally and vertically. An IndexArray is proposed to make use of BitTable horizontally. To find out the representative items of frequent-1 itemsets quickly subsume index is computed by using the breadth first search. Then the depth first strategy is used for the resulting itemsets to generate all other frequent itemsets. An index array is an array with size n1, where n1 is the number of frequent-1 itemsets. Each element of the array belongs to a tuple, where item is an item and subsume of an item is the subsume index. The subsume index of an item is obtained by intersecting all the transactions containing that item. The Index-BitTableFI achieves good performance by computing the subsume index i.e frequent itemsets having the support count as representative items can be identified directly by its subsume index.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PPV[20] is the first algorithm to integrate FP-growth method with vertical mining method. Even though it employs candidate generation and test strategy to find frequent itemsets, it has inherent weakness of Apriori-like methods i.e the candidate itemsets of length(K+1) Node-lists are obtained by intersecting the Node-lists of frequent patterns of length K. To overcome this drawback Node-list is changed to N-list[24]. Based on the Nlist data structure a new algorithm prepost is proposed, which can be used to obtain frequent itemsets directly without candidate generation. To store the database prepost adopts a prefix tree structure called PPC-tree. By traversing the PPC-tree in the preorder and postorder manner each node is assigned with PP-code (PrePost code). The PP-codes are arranged in nondecreasing order of their preorder values. The ancestor-descendent relationship of two nodes X1 and X2 is defined as follows.

X_1 is an ancestor of X_2 if and only if X_1.preorder<X_2.preorder and X_1.postorder>X_2.postorder. Each frequent item can be represented by an N-list, where N-list is the sequence of PP-codes. The N-list of an 'i_1' is $<(a_{1m},b_{1m}):c_{1m}>$ and the N-list of 'i_2' is $<(a_{2n},b_{2n}):c_{2n}>$, the N-list 'i_1i_2' is $<(a_{1m},b_{1m}):c_{2n}>$ if and only if $<(a_{1m},b_{1m}):c_{1m}>$ is an ancestor of $<(a_{2n},b_{2n}):c_{2n}>$. By intersecting the N-lists of frequent-K itemsets prepost finds frequent-(K+1) itemsets.

Let <(a1m,b1m):c1m> and <(a2n,b2n):c2n> be the PP-codes, then the NodeList-intersection is worked as follows.

First check the ancestor-descendent relationship of $<(a_{1m},b_{1m}):c_{1m}>$ and $<(a_{2n},b_{2n}):c_{2n}>$. if $<(a_{1m},b_{1m}):c_{1m}>$is an ancestor of $<(a_{2n},b_{2n}):c_{2n}>$, then check there exists a node in the form of $<(a_{1m},b_{1m}):c_{mn}>$ in the N-list of P. if so, change $<(a_{1m},b_{1m}):c_{mn}>$ to $<(a_{1m},b_{1m}):c_{mn}+c_{2n}>$ otherwise insert $<(a_{1m},b_{1m}):c_{2n}>$ into the N-list of P. Then if $<(a_{2n+1},b_{2n+1}):c_{2n+1}>$ is not null, then check the ancestor-descendent relationship of $<(a_{1m},b_{1m}):c_{1m}>$ and $<(<(a_{2n+1},b_{2n+1}):c_{2n+1}>$.

When compared to Node-lists, N-lists have two advantages. First the length of the N-list of an item is much smaller than the length of the Node-list of an item and the second is N-list follow the single path property. The major advantages of the prepost algorithm are

1) For the compact representation of original database it uses a data structure N-list, which avoids the iterative database, scans in the subsequent mining process
2) An efficient strategy intersection of two N-lists is used instead of counting the support count of itemsets.
3) Without generating the candidate itemsets it finds frequent itemsets by the use of single path property.

The prepost outperforms several algorithms like FP-growth, FP-growth*,Eclat and dEclat in the point of running time. However, it consumes more memory for the sparse datasets than FP-growth and FP-growth*. The representation of PPC-tree is more memory consumed because each node of the PPC-tree contains more information like preorder code, postorder code, and count etc. So it takes more memory than FP-tree.

To improve the efficiency of prepost Bay Vo, et al. present an improved version of prepost [27]. To enhance the process of creating the N-lists associated with itemsets it uses it uses hash table representation and an improved N-list intersection method. First construct the PPC-tree for the given dataset. Each node N of the PPC-tree maintains five values like name of the item, frequency of the item, number of child nodes, preorder code and postorder code. The N-list of a frequent-1 itemset is the sequence of PP-codes associated for that node in the PPC-tree.

Suppose Y_M and Y_N be the two (K-1) itemsets with the same frequency Y. $NL(Y_M)$ and $NL(Y_N)$ are the N-lists associated with Y_M and Y_N. Then the N-list associated with Y_{MN} is fi750nd as follows. For each PP-code $C_i \varepsilon NL(Y_M)$ and $C_j \varepsilon NL(Y_N)$ the algorithm will add <(C_i.preorder, C_i.postorder): C_j.frequency> to $NL(Y_{MN})$ if C_i is an ancestor of C_j. To combine PP-codes which has same preorder and postoder values, traverse $NL(Y_{MN})$. The N-list intersection function [24] was O(x+y+z) where x, y, and z are the lengths of resulting N-list, but the improve N-list intersection method is only O(x+y). Another advantage is to merge the same PP-codes improved intersection function does not traverse the resulting N-list. Subsume index of frequent-1 itemsets were determined to find the representative items, it leads to great reduction in run time. The proposed algorithm is faster than prepost for the dense datasets.

Node-list[20] and N-list[24][27][32] are the two data structures used for efficient mining of frequent itemsets, but these two structures use the PPC-tree with preorder code and postorder code, so

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

these two suffers with the problem of more memory consumption to mine frequent itemsets. To overcome this drawback an efficient data structure named Nodeset[28] was introduced by Zhi-Hong Deng et al. POC- tree is used to represent the transactional dataset. For each node of POC-tree the Nodeset maintains only the preorder code or postorder code, which reduces half of the memory required when compared to Node-lists and N-lists. Based on Nodesets, a new algorithm named Frequent Itemset Mining (FIM) is proposed for efficient mining of frequent itemsets. A search tree called set-enumeration tree is used by FIM for discovery of frequent itemsets. To reduce the search space, it adopts a pruning strategy called promotion, which is similar to children-parent equivalence pruning. The M-info for a node M in a POC tree is, it'spreorder code or postorder code and the count of that item registering in it. The Nodesetof a frequent item A is the sequence of all the M-infos of nodes registering A in the POC tree. Given a frequent item A, assume it Nodeset is {(P1:C1), (P2:C2),……., (Pm:Cm)} where P is the preorder or postorder value and C is its Count. Then the support of an item A is C1+C2+….+Cm. Experimental results shown that the Nodeset structure is effective because of

a) Less memory consumption
b) Fin runs faster than prepost and FP-growth because of reduced search space with pruning strategy.

The improved version of prepost was N-list and Subsume based algorithm for mining Frequent Itemsets (NSFI) [32]. The NSFI uses the hash table to enhance the process of creating N-lists and an improved N-list intersection algorithm. Here two theorems are proposed, to determine the subsume index of frequent-1 itemsets based on the N-list concept. It uses an early abandon strategy which consists of 3 steps.

1) By summing the frequencies of first and second N-lists, total frequency of two Nlists is determined.
2) For each PP-code Cj that does not belong to the resultant N-list TF=TF-Cj frequency.
3) If TF is less than minimum support then stop the process and consider that itemset as infrequent.

The subsume concept [23] was adopted in NSFI algorithm in order to reduce the memory consumption requirements, because it is not essential to store the N-lists associated with a set of frequent itemsets to calculate their supports. The NSFI is faster than prepost for the dense datasets and the runtime of NSFI is always faster than dEclat.

6. COMPARATIVE STUDY OF DIFFERENT FREQUENT PATTERN MINING METHODS

S. No.	Method	Pros	Corns
1	Agrawal Imielinski Swami (AIS) [2]	-When the database is scanned, candidate itemsets are generated and counted on-the-fly basis.	-Unnecessarily generating and counting too many candidates. -Requires multiples scans over the database
2	SETM [2]	- Standard SQL join operator is used for candidate generation. - Support count of itemset is calculated at the end of pass.	- Requires multiple database scans. - For each candidate itemset many entries ass it's support value.
3	Apriori [5]	- Reduces the number of candidates generated when compared to brute-force method. - It is based on candidate-generation and test approach.	- If the database is too large, it requires more time to scan the database for support counting.
4	Apriori Tid [5]	- After the first pass transactional database is not used for support counting. - AprioriTid outperforms Apriori for larger value of 'K'.	- It requires more memory.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5	Improved Apriori Algorithm (IAA) [18]	- Reduces the number of frequent itemsets generated. - The amount of time required for scanning is reduced. Hence, the execution time is reduced. - Performance is improved.	- If the database is too large it takes more time.
6	Transaction Reduction [22]	- Attribute Size Of Transaction (SOT) is used to update the database. - It reduces the total I/O time required and the total time required for scanning.	- It has a overhead of managing the database after updating the database every time. - More memory is required to manage updatable database.
7	Transaction Reduction – Bit Array Matrix(TR – BAM) [31]	- The entire database is scanned only once. - Bit Array Matrix is used to represent the data. - Repetition Count (RC) column is used to represent the repeated data. - Number of nonzero elements in each column is represented with sum.	- More memory consumption.
8	Count Table FI (CTFI) [30]	- Count is used to represent the multiple transactions having the same set of items. - No need to generate candidate, itemsets. So, it reduces costly database scans. - Less execution time and less memory usage.	- Transactional database is represented in the form of a binary number and decimal number which takes more time for conversion.
9	BE-Apriori [29]	- Number of frequent itemsets becomes much less. So, there is a significant reduction in running time. - Transactional database is compressed to reduce the number of transactions to be scanned by maintaining the frequency of all the items in frequent itemsets.	- Overhead of maintaining the temporary table.
10	Frequent Pattern Growth (FP-Growth) [7]	- Two scans are enough to find the frequent itemsets. - It does not generate candidate itemsets. - It reduces the amount of memory required for storing the transactions.	- It generates more number of conditional FP-trees. - If the database is too large, it is difficult to construct a main memory based FP-tree.
11	Diffset [8]	- Rather than maintaining the entire database, it only maintains the differences in Tid's. So, this method greatly reduces the amount of memory required by order of magnitude.	- Number of comparisions are required to find the differences in Tid's, so it requires more execution time.
12	Transaction Mapping (TM) [9]	-Transaction tree is constructed for all Transaction Id's and those are compressed into transaction intervals. This compression reduces the intersection time.	- Need to maintain transaction tree.
13	Bit Table FI [16]	-It uses a special data structure called Bit Table to represent the database which is much smaller in size when compared to the original database. - Bitwise AND operation is used to generate the frequent itemsets, it is much faster than traditional candidate itemset generation procedure.	- It cannot reduce the number of database scans. - It cannot reduce the size of the candidate itemsets.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

14	PrePostVertical (PPV) [20]	- It uses a special tree called PrePostCode-tree (PPC-tree) to store the database. - The Node-list is more effective because transactions with common prefix share the same nodes.	- More memory consumption because each node in a PPC-tree maintains name of the item, frequency of the item, number of child nodes, preorder code and postorder code.
15	BitApriori [21]	- For candidate generation, it scans the database only twice. - It traverse the trie only once for support counting.	- If the trie has many nodes from the root node, it suffers with the problem of memory scarcity.
16	Index –BitTableFI [23]	- It uses the Bit Table horizontally and vertically. - Subsume Index was used to find the representative items.	- Transactional database is transformed into Index Array which takes more time.
17	PrePost [24]	- Frequent itemsets are obtained without candidate generation. - It uses a special data structure named N-list which avoids iterative database scans.	- More memory consumption because each node in a PPC-tree maintains name of the item, frequency of the item, number of child nodes, preorder code and postorder code.
18	Node set [28]	- It represents data in the form of a POC tree i.e either preorder code or postorder code. - It reduces the search space with the special pruning strategy named Promotion.	- More memory consumption if the tree has more nodes.

7. OPEN RESEARCH ISSUES

Association rule mining performing vital role in the essential area data mining. Applications of association rule mining are Large and Distributed database - Businesses, e.g. logistics, marketing and Government - almost all branches e.g. defence, public safety, Spatial database - GIS, Relational database - Industries, Medical database- Medical diagnosis, Hospital, Medical shops, scan centers. Future work is to find out the better algorithm to find frequent itemsets.

8. CONCLUSION

One of the important data mining techniques is association rule mining. Association rule mining is not only used to discover interesting relationships, it also used to discover differences between different kinds of classes in a database. Association rule mining algorithms are basically categorized into two groups named as horizontal mining algorithms and vertical mining algorithms. Horizontal mining algorithm suffers with the problem of repeated scans of database and more number of candidate itemsets generated, where the vertical mining algorithms gave some improvement over the horizontal mining algorithms by reducing the number of scans and by reducing the number of candidate itemsets generated. This improvement leads to significant reduction in run time. This paper briefly gives the overview of some existing frequent pattern mining algorithms. This analysis states that all the methods have its own pros and corns.

REFRENCES:

[1] Han.J, Kamber.M, "Data Mining: Concepts and Techniques", Morgan kaufmann Publishers, Book, 2000.

[2] R. Agrawal, T. Imielinski, A. Swami, "Mining associations between sets of items in large databases, Proceedings of the ACM SIGMOD 1993 Conference Washington DC, USA, May 1993.

[3] R. Srikant and R. Agarwal, " Mining quantitative association rules in large relation tables" proceedings of the 1996 SIGMOD, pp. 1-12, 1996.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[4] T. Karthikeyan and N. RaviKumar, "A Survey on Association rule mining", International Journal of Advanced Research in Computer and Communication Engg(IJARCCE), PP.5223-5227, 2014.

[5] R. Agarwal and R. Srikant, " Fast algorithm for mining association rules", Proceedings of the 20th international conference on very large databases , Margunkaufmann , PP. 487-499.

[6] J. Han, M.Kamber, "Data Mining: Concepts and Techniques", The Morgan kaufmann Series in Data Management Systems, Champaign: CS 497JH, fall 2001.

[7] Jiawei Han, Jianpei, and Yiwenyini, "Mininig Frequent Patterns without Candidate Generation", Proceedings of the ACM SIGMOD International Conference on Management of Data Pages , PP. 1-12, 2000.

[8] Mohammed J. Zaki and Karam Gouda, "Fast Vertical Mining using Diffsets" Proceedings of the ASM SIGKDD '03 Washiton, DC, USA, Aug-2003.

[9] Mingjun Song, and SanguthevarRajasekaran, "A transaction mapping algorithm for frequent itemset mining ", in IEEE transactions on knowledge and Data Engg.

[10] M. J. Zaki, S. Parthasarathy, M. Ogihara, and whi, "New Algorithms for Fast Discovery of Association Rules" Proceedings of KDD – 97, pp. 983-286.

[11] Jianyong Wang, Jiawei Han, " TFP: An Efficient Alogirithm for Mining Top-K Frequent Closed Itemsets" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 17, NO. 5, MAY 2005.

[12] M.J. Zaki and C.J Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining." Proc. 2002 SIAM Int' I Conf. Data Mining[SDM '02), pp. 457-473. Apr. 2002.

[13] I. Wang, J. Han, and J. Pei, "CLOSET+ : Searching for the Best Startagesies for Mining Frequently Closed Itemsets, " Prc. 2003 ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining(KDD '03), pp. 236-245, Aug. 2003.

[14] Yew KwongWoon, Wee Keong Ng, Ee-Peng Lim, "A Support-Ordered Trie for Fast Frequent Itemset Discovery", IEEE transactions on Knowledge and Data Engineering , PP. 875-879, Aug. 2004.

[15] Y.K. Woon, W.K,Ng, and A. Das, "Fast Online Dynamic Association Rule Mining", Proc. Secont Int'l Copnf. Web Informatoion Systems Eng., pp. 278-287, 2001.

[16] Jie Dong, Min Han "BitTableFI: An efficient mining frequent itemsets algorithm", Knowlwdge-Based Systems, vol.20, pp.329-335, 2007.

[17] D.Burdick, M. Calimlim, J. Flannick, J. Gehrke, T.M. Yiu. "MAFIA: a maximal frequent itemset algorithm", IEEE Transactions on Knowledge and Data Engineering 17(11) (2005) 1490-1504.

[18] Huan Wu, Zhigang Lu, Lin Pan, RongshengXu, "An Improved Apriori-based Algorithm for Association Rules Mining", Sixth International Conference on Fuzzy Systems and Knowledge Discovery, pp. 51-55, 2009.

[19] Lei Ji, Baowen Zhang, and Jianhua Li, "A New Improvement on Apirori Algorithm", Computational Intelligence and Security, 2006 International Conference on Volume 1, Nov 2006, pp 840-844.

[20] Zhihong Deng, Zhonghui Wang, " A New Fast Vertical Methof for Mining Frequent Patterns" International Journal of Computational Intelligence Systems, Vol 3, No. 6,PP. 733-744, Dec 2010.

[21] JieminZheng, Defu Zhang, Stephen C. H. Leung, Xiyue Zhou, "An efficient algorithm for frequent itemsets in data mining", International Conference on Advances in Signal Processing and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Communication (ICSSSM), PP. 1-6, June 2010.

[22] Jaishree Singh, Hari Ram, Dr. J.S.Sodhi, " Improving Efficiency of Apriori Algorithm Using Transaction Reduction", International Journal of Scientific and Research Publications, Vol.3, 2013.

[23] Wei Song, Bingru Yang, ZhangyanXu, " Index-BiTTableFI: An improved algorithm for mining frequent itemsets" Knowledge-Based Systems, Vol. 21,PP. 507-513, 2008.

[24] Deng. Z., Wang. Z., & Jiang, J.(2012). "A new algorithm for fast mining frequent itemsets using N-lists" , SCIENCE CHINA Information Sciences , Vol. 55, PP. 2008-2030, 2012.

[25] Zubair Khan, NeetuFaujdar, Prashantkumar sing, Tarifabbas , " Modified Bit Aopriori Algorithm: An Intelligent Approach for Mining Frequent Item-Set" Proc of Int. Conf. on Advances in Signal Processing and Communication, PP.813-819, 2013.

[26] V. Vijayalakshmi, Dr. A Pethalakshmi, "Mining of Frequent Itemsets with an Enhanced Apriori Algorithm" International Journal of Computer Applications(0975-8887) Volume 81 – No. 4. November 2013.

[27] Bay Vo, Tuong Le, FransCoenen, Tzung-Pei Hong , " A Hybrid approach for Mining Frequent Itemsets" Systems, Man, and Cybernatics, IEEE, PP.4647-4651, 2013.

[28] Zhi-Hong Deng, Sheng-Long Lv, " Fast mining frequent itemsets using Nodesets" Expert Systems with Applications 41 (2014) 4505- 4512.

[29] Zhuang Chen, Shiban Cai, Qiulin Song, and Chonglai Zhu, " An Improved Apriori Algorithm Based on Pruning Optimization and Transaction Reduction", Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), PP. 1908-19011, Aug-2011.

[30] Marghny .H, Mohamed .M, and Darwieesh, " Efficient Mining Frequent Itemset Algorithms ", International Journal of Machine Learninbg and Cybernatics, Vol. 5, PP. 823-833, 2013.

[31] V. Vijayalakshmi, Dr. A Pethalakshmi, "An Efficient Count Based Transaction Reduction Approach For Mining Frequent Patterns", Procedia Computer Science, Vol.47, PP. 52-61, 2015.

[32] Bay Vo, Tuong Le, Frans Coenen, and Tzung-pei Hong, "Mining frequent itemsets using the N-list and subsume concepts", International Journal of Machine Learning & Cybernatics, PP. 1-13, Apr 2014.