A Pragmatic Approach on Association Rule Mining and its Effective Utilization in Large Databases

Biswaranjan Nayak [1], Srinivas Prasad [2]

[1] Dept. Of Computer Science Engineering, Trident Academy of Technology
Bhubaneswar, Odisha- 751024, India

[2] Dept. of Computer Science Engineering, Gandhi Institute for Technological Advancement
Bhubaneswar, Odisha-752054, India

Abstract

This paper deals with the effective utilization of association rule mining algorithms in large databases used for especially business organizations where the amount of transactions and items plays a crucial role for decision making. Frequent item-set generation and the creation of strong association rules from the frequent item-set patterns are the two basic steps in association rule mining. We have taken suitable illustration of market basket data for generating different item-set frequent patterns and association rule generation through this frequent pattern by the help of Apriori Algorithm and taken the same illustration for FP-Growth association rule mining and a FP-Growth Tree has been constructed for frequent item-set generation and from that strong association rules have been created. For performance study of Apriori and FP-Tree algorithms, experiments have been performed. The customer purchase behaviour i.e. seen in the food outlet environments is mimicked in these transactions. By using the synthetic data generation process, the observations has been plotted in the graphs by taking minimum support count with respect to execution time. From the graphs it has that as the minimum support values decrease, the execution times algorithms increase exponentially which is happened due to decrease in the minimum support threshold values make the number of item-sets in the output to be exponentially increased. It has been established from the graphs that the performance of FP-Growth is better than Apriori algorithm for all problem sizes with factor 2 for high minimum support values to very low level support magnitude.

Key words: *Association Rule Mining, Confidence, Support, Data Mining*

1. Introduction

Data mining which in other way also referred as knowledge discovery process, basically aims at discovering interesting correlations, patterns and trends exist among data items in large data bases. So, the patterns or association and trends discovered in the data mining process provide a helping hand for decision making process in its management information system for better guidance in the wake up of business benefit for the organization. For example, a super market chain company which sells different items everyday in its outlets situated in all over the world. Suppose the manager finds out that most of the time, bread, butter and milk are sold more than other food stocks so that a dearth of non-availability of these items arises frequently.

It seems that as the number of customers is more to buy but due to non-availability of items the outlet loses its business revenue.

Not only that there may be other some items which are also sold frequently. Again there are customers, when they buy particular one or more items; simultaneously they purchase some other items positively. For example, customers who buy bread and butter they also purchase milk or egg definitely. So the manager of the super market outlet wants to know which items are frequently sold out so that he can able to stock more amount of those items so that customers will get the chance to purchase those items. Also for more business attraction the management can think for giving some discounts on the products which will again attract more customers to come for buying those items from the super market boosting the high sales target of the super market outlet.

For having an idea about association rule mining techniques and its effectiveness that play a major role in business transaction process, let us discuss in detail.

Basically, association rule mining is a data mining method which formulates correlation among a set of items with other set of items in a database which was first proposed by three researchers named R. Agrawal, T. Imielinski and A.N. Swami in 1993 [18]. The advantages of these rules

are that they can be utilized for extracting exhilarating correlations, frequent association, patterns or clausal structures among sets of items that are present in the large transactional databases or any additional repositories [10]. Association rule mining covers wide application areas spanning from market-basket database used in big super market applications, telecommunication networks, risk management, financial investment, manufacturing and production, scientific application domains, health care to World Wide Web [14] where large volumes of data are stored either in centralized databases or in distributed databases.

Association rule mining try to find out the items in the database which are expected to be associated together with an interesting relationship so that it will pave the way for execution of effective decision making process of the organization while dealing huge set of transactional business records.

2. Concepts of association rule mining

It is one of the most well-liked data mining techniques presented in [18] where different patterns discovered using this technique is represented through different set of association rules. It can be better understood as described below.

Suppose, I known as an Item-set, represents a set of items sold by a store. So,

$I = \{i_1, i_2, i_3, i_4, \ldots\ldots, i_m\}$

I_k is known as data-item such that $1 \leq k \leq m$.

Total numbers of elements present in the item-set I, specified as the length of data-item.

$\mathcal{D}$ = a database of different transaction records and

$\mathcal{T}$ =transaction that contains a set of items in such a manner so that $\mathcal{T} \subseteq I$.

Keeping all this together, in the transaction database $\mathcal{D}$, an association rule can be expressed statistically in the form $X \rightarrow Y$, where $X, Y \subseteq I$ and $X \cap Y = \emptyset$.

That means, the member satisfy the conditions in X must have to satisfy the conditions in Y. Here, X is known as antecedent and Y is known as consequent.

In other way: A1 ∧.....∧Am=> B1∧....∧ Bn.

Where, ($i \in \{1,...,m\}$) for Ai and ($j \in \{1,...,n\}$) for Bj and both Ai & Bj are called pair of attribute-value pair. It clearly represents that data $B1,\ldots.., Bj$ is pertinently going to appear along with data $A1, ..., Ai$. Hence, the association rule depicts the total number of occurrence of Y with respect to occurrence of X already happened depending on the support and confidence value which are the two important basic measures for association rules.

Support:

It establishes the importance of association rule which is calculated as the probability of item or item-sets in the transactional database and represented in the form:

$$support(X) = n(X)/n$$

Here, n(X) represents number of transactions containing the item-set X and n represents total number of transactions containing in the database $\mathcal{D}$. If it is said that the support of an item is 0.06%, then it is understood that only 6 percent of this items are purchased in the transaction process. If the support value is high, the rule is more important and it will be used widely.

Confidence:

It is calculated as the conditional probability of item-set Y in transactions of item-set X. It is represented in the form:

$$confidence(X \rightarrow Y) = support(X \cup Y)/support(X)$$

This means, it is the percentage calculation of number of transactions of both X and Y divided by the number of transactions of X. If it is said that confidence of the association rule $X \rightarrow Y$ is 80%, then it is understood that in 80 percent of the transactions both X and Y contain together.

The job of association rule mining is to find association rules that satisfy two things; the predefined minimum support and confidence from a given database. Data mining association rules is generally divided into two sub-problems:

i) Minimum support and large item-sets/frequent item-sets.
ii) Minimum confidence and frequent item-sets.

In the first case, those item-sets which possess support value more or equal to minimal support value are found outs. The item-sets are treated as large item-sets whose support values surpass the minimum support vale and those item-sets that are expected or have the hope to be large are called frequent item- sets. But in the second case, it generates all association rules from those large item-sets with a minimum confidence value. The item-sets are known as candidate sets.

So, in the whole process, whenever all the large item-sets are revealed, it helps to obtain preferred association rules.

3. Illustration of association rule mining corresponding to prominent association rule mining algorithms (Apriori & Fp-Growth)

The item-sets of length m are referred as m-item-sets. Among different techniques, frequent item-set generation

is prominent one for an association rule mining purposes. In fig. 1 frequent item-set generation has been depicted in three steps.

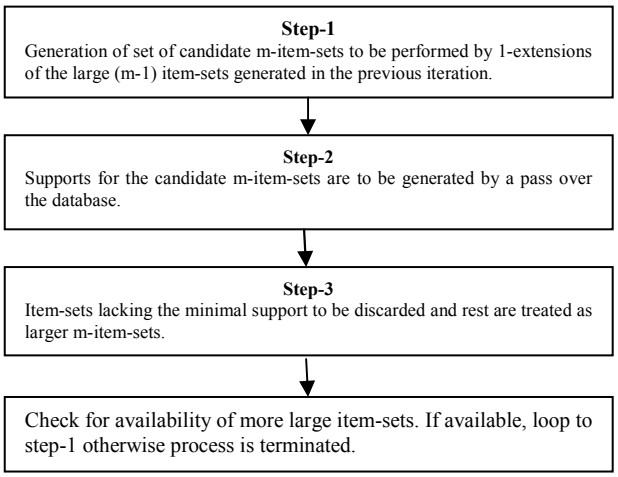

Fig. 1 Fundamental steps for frequent item-set generation

To have a thorough understanding about association rule mining,, here we have taken market basket data analysis as an illustration. A market basket database stores different food items such as bread, butter, milk, egg, sauce etc. that are sold in the food outlet. Suppose, the manager wants to know those items, which are purchased by customers together or they are interesting to purchase those items together so that he may keep the optimum stock of those items for high sale and profit. Not only that, if the manager wants to offer some discount on certain items for promoting more sale then how he will take a decision upon this? Though there are more than thirty algorithms, here we have taken two prominent algorithms called Apriori and FP-Growth for our market basket analysis illustration.

3.1 Apriori Algorithm

Apriori algorithm is one of the prominent multi-pass association rule mining algorithm which deals with frequent item-sets. The overview of the algorithm is depicted in the fig. 2.

In the first pass of the algorithm, item occurrences are counted to determine the large l-item-sets. A succeeding pass, for instance pass k, is comprised of two phases. In the first phase, the large item-sets Lk-1 that are found in the(k-1)th pass are used for generating the candidate item-sets Ck, through the Apriori candidate generation function which is described in fig. 1. In the second phase, scanning for database is performed and counting for support of candidates in Ck is done. For speedy counting, an efficient determination is desired when the candidates in Ck are contained in a given transaction t. A hash tree data structure as described in [9] is utilized for this purpose.

```
L1 = {large 1-itemsets};
for ( k = 2; Lk-1 ≠ Ø; k++ ) do begin
Ck = apriori-gen(Lk-1); //New candidates
forall transactions t ∈ D do begin
Ct = subset(Ck, t);
//Candidates contained in t
forall candidates c ∈ Ct do
c.count++;
end
Lk = { c ∈ Ck | c.count ≥ minsup }
end
Return ∪k Lk;
```

Fig. 2 Pseudo code of apriori algorithm

Now we will analyze the Apriori algorithm with respect to market basket taking 10 transactions for discovering those items that are frequently associated together in the series of transactions occurred. In the market basket example following parameters are considered.

- A database $\mathcal{D}$ is considered which is comprised of 10 transactions.
- Minimum support count is taken as 2 for instance because min-sup= 2/10= 20%.
- Let the required minimum confidence is 70%.

From 100000 transactions in a database suppose following 10 transactions are randomly selected for our illustration as shown in table 1.

Table 1
Example of selected 10 transactions randomly from database

Transaction ID	Listed Items
T00001	Bread, Butter, Milk
T00002	Bread, Sauce
T00003	Butter, Egg
T00004	Bread, Butter, Sauce
T00005	Bread, Egg
T00006	Butter, Egg
T00007	Bread, Egg
T00008	Bread, Butter, Egg, Milk
T00009	Bread, Butter, Egg
T00010	Bread, Egg

As, it has been stated earlier that, association rule mining will be done in two different phases, in the first phase frequent item-sets are required to be found out like 1 item-set generation, 2 item-set generation, till all the frequent item-sets are generated with respect to fulfilling minimum support.

First Step: Generating 1 item-set frequent pattern.

Here individual items (candidates) from the database $\mathcal{D}$ are counted and listed as shown in the table 2. Each item becomes a member of the set of candidate, in the first iteration of the algorithm.

Table 2
Generated 1item-sets frequent patterns with support count

Item-sets	Support-Count
{Bread}	7
{Butter}	7
{Milk}	2
{Sauce}	2
{Egg}	7

Table 3
1 item-sets frequent patterns with their support count (satisfying minimum support count condition)

Item-sets	Support-Count
{Bread}	7
{Butter}	7
{Milk}	2
{Sauce}	2
{Egg}	7

Once, the list is made, and then a comparison is made between candidate support count and minimum support count. Those items whose support counts are less than minimum support count are removed and remaining candidate items in the item-set (table 3) are joined for generation of 2 item-set frequent pattern.

Second Step: Generating 2 item-set frequent patterns.

In this step, the algorithm uses joining process of items to generate a 2 item-set frequent patter as displayed in the table 4.

Table 4
Generated 2 item-sets frequent patterns

Item-sets
{Bread, Butter}
{Bread, Milk}
{Bread, Sauce}
{Bread, Egg}
{Butter, Milk}
{Butter, Sauce}
{Butter, Egg}
{Milk, Sauce}
{Milk, Egg}
{Sauce, Egg}

Then the database $\mathcal{D}$ is scanned and support count for each of 2 item-set is counted as shown in table 5.

Table 5
Generated 2 item-sets frequent patterns with support count

Item-sets	Support Count
{Bread, Butter}	4
{Bread, Milk}	2
{Bread, Sauce}	1
{Bread, Egg}	5
{Butter, Milk}	2
{Butter, Sauce}	2
{Butter, Egg}	4
{Milk, Sauce}	0
{Milk, Egg}	1
{Sauce, Egg}	0

Then, again comparison made between 2 candidate support count and minimum support count. Those frequent 2 item-set satisfy the minimum support condition is presented in table 6.

.

Table 6
2 item-sets frequent patterns with their support count (satisfying minimum support count condition)

Item-sets	Support Count
{Bread, Butter}	4
{Bread, Milk}	2
{Bread, Egg}	5
{Butter, Milk}	2
{Butter, Sauce}	2
{Butter, Egg}	4

Third Step: Generating 3 item-set frequent patterns.

In the 1st and second step Apriori property has been not used. To generate 3 item-set frequent pattern Apriori property [1] is used by using join procedure as shown in table 7.

Table 7
Generated 3 item-sets frequent patterns

Item-sets
{Bread, Butte, Milk}
{Bread, Butter, Egg}
{Bread, Butter, Sauce}
{Butter, Milk, Sauce}
{Butter, Sauce, Egg}
{Butter, Egg, Milk}

Once join procedure for 3 item-set frequent patterns is completed prune step is required to be used for reduction the size of the 3 item-set so that heavy computation of large Ck can be avoided. According to Apriori property [9], all subsets of a frequent item-sets is required to be frequent also, it can be determined that four latter candidates possibly can not be frequent which can be understood with the following example.

Suppose we will take {Bread, Butter, Milk} whose 2 item subsets are {Bread, Butter}, {Bread, Milk} and {Butter, Milk}. Now it is found that all 2 item subsets of {Bread, Butter, Milk} are the member of 2 item-set frequent pattern (refer to table 6), we will keep {Bread, Butter, Milk} in 3 item-set frequent patter.

Now if another 3 item-set {Bread, Butter, Sauce} is taken, whose 2 item subsets are {Bread, Butter}, {Bread, Sauce} and {Butter, Sauce} then it is observed that {Bread, Sauce} is not a member of 2 item-set (refer to table VI) which means it is not frequent. Therefore {Bread, Butter, Sauce} is required to be removed from 3 item-set. This shows the pruning procedure. Like this all the generated 3 item-sets are required to be verified for its subsets are frequent or not.

After checking for all members and performing the pruning procedure the transactions in database $\mathcal{D}$ is

scanned for determining the 3 item-set candidates satisfying the minimum support count condition and only two sets are selected as displayed in table 8.

Table 8
3 item-sets frequent patterns with their support count (satisfying minimum support count condition)

Item-sets	Support Count
{Bread, Butte, Milk}	2
{Bread, Butter, Egg}	2

Fourth Step: Generating 4 item-set frequent patterns.

The Apriori algorithm uses join procedure using 3 item-set frequent patterns for generating 4 item-set frequent pattern which produce the set shown in table 9. But again according to Apriori property [9], its subset {butter, Milk, Egg} has been already pruned as it is not frequent. (refer to table 8).

Table 9
4 item-sets frequent patterns

Item-set
{Bread, Butte, Milk, Egg}

As generated 4 item-set frequent pattern = Φ, the algorithm is terminated here after finding all the frequent item-set. So first phase is completed and in the second phase these frequent item-sets will be utilized for generating association rules that will satisfy both minimum support and minimum confidence criteria.

Fifth Step: Generating strong association rule by using frequent item-sets.

According to [9] the following two steps are required to augment the association rule generation.

i) For every frequent item-set "I", all non-empty subsets of "I" is required to be generated.

ii) For all non-empty subset***s*** of ***l***, if support_count(l) / support_count(s) >= min_conf (min_conf= minimum confidence threshold) then output the rule as "**s → (l-s)".**

Now, in our market basket example, item-set L is represented as:
L= {{Bread}, {Butter}, {Milk}, {Sauce}, {Egg}, {Bread, Butter}, {Bread, Milk}, {Bread, Egg}, {Butter, Milk}, {Butter, Sauce}, {Butter, Egg}, {Bread, Butter, Milk}, {Bread, Butter, Egg}}

Suppose we will take frequent item-set I = {Bread, Butter, Milk} which has following non-empty subsets.
{Bread, Milk}, {Bread, Butter}, {Butter, Milk}, {Bread}, {Milk}, {Butter}

In the beginning it has taken minimum confidence level to 70% (i.e. minimum threshold level). Therefore, the rule whose minimum confidence is not less than minimum confidence level, those association rules is selected and others are rejected. As we have two 3 item-set frequent pattern i.e. {Bread, Butter, Milk} and {Bread, Butter, Egg}, here {Bread, Butter, Milk} has been taken for association rule generation. The generated list is shown in table 10. From the table, three strong association rules has been obtained.

Table 10
List of Association rule generated from
Frequent patterns item-sets with selection/rejection

Rules	Confidence (calculated with support count)	Rule Selection/ Rejection
R1: Bread & Milk → Butter	sc{Bread, Butter, Milk}/sc{Bread, Milk}= 2/2=100%	Selected
R2: Milk & Butter → Bread	sc{Bread, Butter, Milk}/sc{Milk, Butter}= 2/2=100%	Selected
R3: Bread & Butter → Milk	sc{Bread, Butter, Milk}/sc{Bread, Butter}= 2/4=50%	Rejected
R4: Bread → Milk & Butter	sc{Bread, Butter, Milk}/sc{Bread}= 2/7= 28%	Rejected
R5: Milk → Bread & Butter	sc{Bread, Butter, Milk}/sc{Milk}= 2/2=100%	Selected
R6: Butter → Bread & Milk	sc{Bread, Butter, Milk}/sc{Butter}= 2/7=28%	Rejected

Also by taking {Bread, Butter, Egg} item-set list of association rule can be generated as above.

3.2 The FP-Tree/Growth Algorithm

It is proposed in [12] where, after a preprocessing scan over the database, it constructs a condensed representation of the database which is known as FP-tree and then data mining is performed over the FP-tree. The overview of the algorithms is described in fig. 3.

Input: A transactional database *DB* and a minimum support threshold ξ.
Output: Its frequent pattern tree, **FP-tree**
Method: The **FP-tree** is constructed in the following steps:

1. Scan the transaction database *DB* once. Collect the set of frequent items *F* and their supports. Sort *F* in support descending order as *L*, the *list* of frequent items.
2. Create the root of an **FP-tree**, *T*, and label it as "root". For each transaction *Trans* in *DB* do the following.
   a. Select and sort the frequent items in *Trans* according to the order of *L*. Let the sorted frequent item list in *Trans* be [*p* | *P*], where *p* is the first element and *P* is the remaining list. Call *insert_tree*([*p* | *P*], *T*).
   b. The function *insert_tree*([*p* | *P*], *T*) is performed as follows. If *T* has a child *N* such that *N.item-name* = *p.item-name*, then increment *N*'s count by 1; else create a new node *N*, and let its count be 1, its parent link be linked to *T*, and its node-link be linked to the nodes with the same *item-name* via the node-link structure. If *P* is nonempty, call *insert_tree*(*P*, *N*) recursively.

Fig. 3 Algorithm for FP-tree construction

Now we will analyze the FP-Tree algorithm by taking the market basket illustration with the same ten transactions from the database $\mathcal{D}$ (refer to table 1). The different criteria have been taken as:

i) 10 number of same transaction records.
ii) Minimum support count required =2 i.e. (2/10=20%)
iii) Here, the database is scanned first just like it is done in Apriori algorithm, i.e. set of 1-itemsets and their respective support counts are found out.
iv) But, the generated frequent item-sets are sorted according to descending order of support count.

Now, the resulting descending ordered set is represented in the table 11.

Table 11
Descending ordered list of 1 item-sets frequent patterns with support count

Item-sets	Support-Count
{Bread}	7
{Butter}	7
{Egg}	7
{Sauce}	2
{Milk}	2

FP-Growth method is done in two phases, i.e.

i) Constructing FP-Tree.
ii) Mining the tree by creating conditional (sub) pattern bases.

i) Construction of FP-Tree

The following steps are required to construct the FP-Tree [9].

- Create first, the root of the tree and labeled as "Null".
- The database $\mathcal{D}$ is scanned for second time.
- The items in every transaction record are processed in a sorted order.
- For each transaction, a branch is created with items having their support count separated through colon symbol (:).
- At any time, if the same node is encountered in another transaction, then the support count of the common node or prefix is incremented.
- For making the tree traversal easier, an item header table is built where each item points to its occurrences in the tree through a chain of node-links is specified.

The constructed FP-Tree is shown in the fig. 4. Once the tree construction is completed, the next phase is to mine the tree.

ii) Mining the tree by creating conditional (sub) pattern bases.

It requires the following steps to be done for mining the tree [9].

- For initial suffix pattern, start from each frequent length-1 pattern.
- It conditional pattern base is constructed by set of prefix paths in the FP-Tree co-occurred with the suffix pattern.
- Then, construct its conditional FP-Tree and mining on that tree is carried out.
- To achieve the pattern growth, suffix patterns are concatenated along with the frequent patterns generated from a conditional FP-Tree.
- The unions of all frequent patterns which are generated using the fourth step provide the required frequent item-set.

By utilizing the above five steps, mining the FP-Tree is performed through creating conditional sub pattern bases which is depicted in the table 12. The explanation of the mining is described now.

Suppose, we have started from the item Milk which is involved in 2 branches i.e. {Bread Butter Milk: 1 and (Bread Butter Egg Milk: 1}. By considering Milk as suffix, the two of its resultant prefix paths are {Bread Butter:1} and (Bread Butter Egg: 1}, which forms the conditional (sub) pattern base for it. We have first found out the support count of individual items in the conditional (sub) pattern base as calculated below:

Support count for Bread= 2 (1+1).
Support count for Butter= 2 (1+1).
Support count for Egg = 1.

As only Bread and Butter satisfy the required minimum support i.e. 2, therefore Bread and Butter items are selected and egg is not selected. So, we have got the conditional FP-Tree now. All frequent pattern related to suffix Milk are generated with the combination of both Milk and conditional FP-Tree. In the same manner, suffixes of Sauce, Egg, and Butter are mined out except for suffix Bread because; it does not possess any prefix. Due to compact data structure, elimination of repeated database scan, no candidate generation or candidate test is not required; therefore FP-Growth is faster than Apriori algorithm.

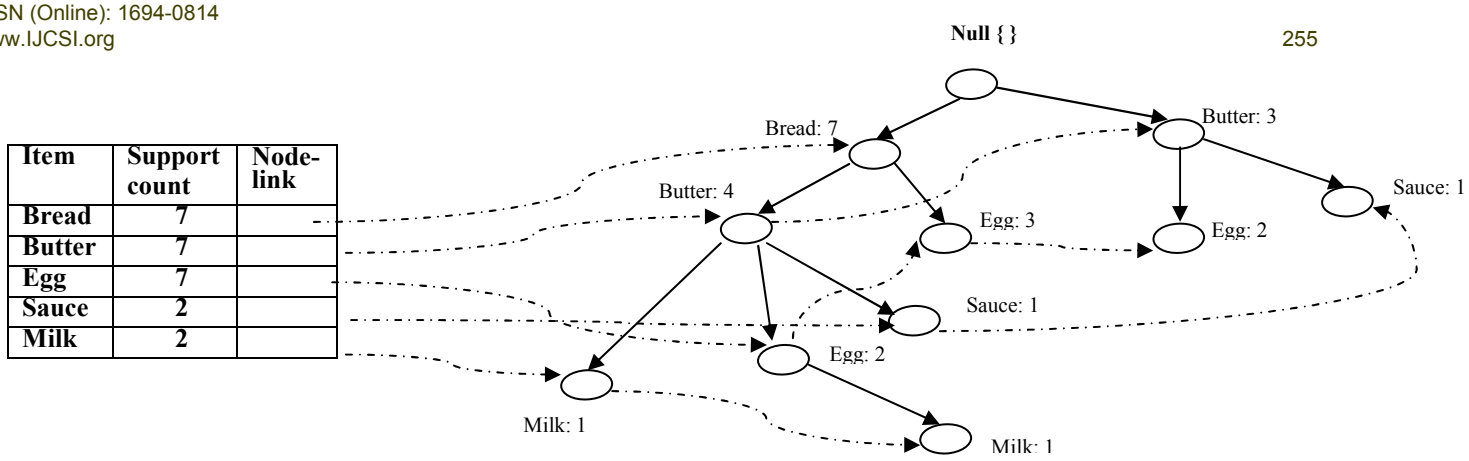

Fig. 4. FP-Tree construction for market basket data analysis

Table 12
Mining of FP-Tree through conditional sub pattern bases

Items	Conditional (sub) pattern base	Conditional FP-Tree	Generation of Frequent Pattern
Milk	{(Bread Butter: 1), (Bread Butter Egg:1)}	<Bread: 2, Butter: 2>	Bread Milk::2, Butter Milk::2, Bread Butter Milk::2
Sauce	{(Bread Butter: 1), (Butter: 1)}	<Butter: 2>	Butter Sauce:2
Egg	{(Bread, Butter:1), (Bread:2)}, (Butter:2)}	<Bread: 3, Butter: 3>	Bread Egg:3 , Butter Egg:3, Bread Butter Egg:2
Butter	{(Bread:2)}	<Bread: 2>	Bread Butter : 2
Bread	**It does not have any prefix, so it is not taken into consideration for mining purpose.**		

4. Related work

Generation of association rule was first introduced in [18] and AIS algorithm was proposed for mining of all types of association rules. An algorithm called SETM was proposed in [19] for solving association rule generation problem by using relational operations. Two new algorithms called Apriori and Apriori Tid were proposed in [17] which were much better in performance as compared to previous algorithms.

The designed algorithms vary according to two factors, i.e. i) generation of candidate item-sets and counting of support for the candidate item-sets. In [18] candidate item-sets are generated with multiple pass over the database. For each of the transaction, the candidate item-sets are generated by such a manner so that the new item-sets are contained in that transaction. But candidate item-sets are generated by using the large item-sets from the previous pass by joining the set of large item-sets with itself in [17]. The resulting candidate set is required to be pruned for elimination of those item-sets whose subset is not enclosed in the previous large item-sets in turn it produces a much smaller candidate set as compared to former technique used in [18]. For each transaction, the supports for the candidate item-sets are found out by identifying all the candidate item-sets included in that transaction and the item-sets are counted by incrementing one.

For sub-set operations, how the data structures were used has not described in [18] whereas data structures used in [9] differs accordingly.

In Apriori, the candidate item-sets are required to be compared with the transactions for determining whether they are included in the transaction or not as well as a hash-tree structure is used for restricting the set of candidate item-sets comparison. It optimizes the subset testing. To make the testing faster bitmaps are used in place of transactions. But in Apriori Tid, after each pass, in place of transaction, an encoding of all the large item-sets included in the transaction is used. In the next pass, a check is done to see whether the large item-sets used for generating the candidate item-sets are contained in the encoding of the transaction or not and then item-sets are tested for inclusion in the transaction. Another key difference between Apriori and Apriori Tid algorithm is that, subset testing is performed for each transaction in every pass in Apriori whereas, in Apriori Tid, if in the current pass, a transaction does not contain any large item-sets then that transaction is not measured for consequent passes. Apart from that, in the afterward passes, the size of the encoding can be found drastically smaller than the actual database.

In DIC Algorithm which is proposed in [16] candidates are generated as well as removed after every m transactions where M is used as a parameter in the algorithm. It utilizes multi-pass technique but typically it can be completed

within two passes. But the drawback is that it also follows a tuple-by-tuple approach.

An algorithm call CARMA is proposed in [13] actually a two pass algorithm that can able to dynamically generate and remove candidates after processing each tuple of the database.

FP-Growth algorithm was proposed in [12] for construction of compact or condensed database through creating FP-Tree and the mining is performed on this FP-Tree.

MaxClique Algorithm is proposed in [15] which is designed to efficiently mine databases that are available in a vertical layout in contrasts to all the above algorithms are primary based on horizontal approaches i.e. tuple manner approaches.

VIPER Algorithm is proposed in [11] which utilizes also vertical approaches without any restrictions whereas earlier vertical mining algorithm had various restrictions on the principal database size, shape contents or the mining process.

In [5], a new algorithm is proposed which is intended to extract the best rules in a realistic time of execution but there is no guaranty of optimal solutions. The algorithm is based on Quantum Swarm Evolutionary approach which yields better results in comparison to genetic algorithms.

In [8], a novel approach for making effective decision tool especially for traffic safety administrators is proposed where crashes are analyzed as supermarket transactions to detect interdependence among crash characteristics. Through the association rules discovered from the analysis it has been observed that there exists a significant correlation between lack of illumination and high severity of crashes as well as results are consistent with the understanding of crash characteristics The advantage is that it helps to build good decision support tool.

In [2], the researchers proposes locality sensitive hashing technique for sampling-based algorithms in association rule mining which can able to process very large databases in a reasonable time. The proposed algorithm obtains more accuracy as well as execution time is less as compared to previously proposed algorithms.

An evolutionary algorithm to discover quantitative association rules from huge databases without the need for an a priori discretization is proposed in [1]. The salient features of the proposed algorithms are i) it is not obligatory to carry out an apriori discretization., ii) the fitness function includes the measures for finding the most significant rules. Apart from these the algorithm can able to obtain overlapping rules as well as it can work under different levels of noise.

In [3], the researchers designed a data structure to generate the association rules between the items at different levels in a taxonomy tree. The key advantage of the proposed algorithm is that, fewer candidate item-sets are generated here. The method effectively prunes a large amount of irrelevant rules that are based on minimum confidence.

In [7], a novel method proposed, called HUINIV (High Utility Item-sets with Negative Item Values)-*Mine*, for efficiently and effectively mining high utility item-sets from large databases by considering negative item values and this concept is probable the first method that deals the negative item values in utility mining. The method can effectively identify high utility item-sets by generating fewer high transaction-weighted utilization item-sets such that the execution time can be reduced significantly in mining the high utility item-sets. Apart from this, the process of discovering all high utility item-sets with consideration of negative item values can be accomplished effectively with less requirements on memory space and CPU I/O which fulfills the critical requirements of temporal and spatial efficiency for mining high utility item-sets with negative item values and generates much less candidate item-sets under different experimental conditions.

In [4], a multi objective association rule mining method is proposed which utilizes genetic algorithm without taking the minimum support and confidence into account. FP-tree algorithm is applied for improving efficiency. In the proposed method which extracts the best rules that have best correlation between support and confidence. The operators used are flexible for changing the fitness and unlike the Apriori-based algorithm it does not depend on support as well as the method outperforms the traditional methods.

In [6], a de-clustering approach for distributed architectures is proposed, which eliminates the inter-site communication cost, for most of the influential association rule mining algorithms where to de-cluster the database into similar partitions, a proficient algorithm is developed to approximate the shortest spanning path (SSP) for bonding the transaction data together. The SSP obtained is then used to evenly de-cluster the transaction data into subgroups. The method guarantees that all subgroups are similar to each other and to the original group. It has been observed in the study that most of the influential association rule mining algorithms can be implemented in a distributed architecture so that without losing any frequent item-set the speed can be increased tremendously.

5. Experimental study for performance measure of Apriori and fp-growth algorithms with synthetic data generation

Here, we have performed experiment for the performance study of two association rule mining algorithms i.e. Apriori and FP-Growth. By obtaining the sales data of a multi-chain company that sell food items, home accessories, dress materials etc, where databases have been synthetically generated for the experiments according

to the techniques followed in [17]. The synthetic generators by using the different parameters and the default values have considered the following datasets named DSET1, DSET2 and DSET3. Each of the dataset possesses the capacity of 10 millions transactional tuples in it. The list of parameters and default values used for the DSET1, DSET2 and DSET3 are shown in the table 13.

Table 13
List of Parameters used for generating Synthetic datasets

Parameters with definition	Default Values assigned
D: Total number of transactions in the database	10M
N: Total number of items	1000
L: Number of prospective item-sets	2000
I: Mean length of prospective frequent Item-sets	4, 8, 12
T: Length of the Average Transaction	10, 20, 40

With the available primary memory, we have set the very low minimal support values for frequent item-sets. It has been observed that the numbers of frequent item-sets are exceeded more than twenty thousands and beyond that, definitely it would be very difficult to find interesting patterns from the mammoth rules being generated. We have set the minimum support threshold values for the three synthetic datasets are ranging from 0.5% to 2% and our experiment run on small databases comprising 100K transactions. The results of this experiment are shown in the Fig. 5, Fig. 6, and Fig. 7 corresponding to three datasets named DSET1, DSET2 and DSET3.

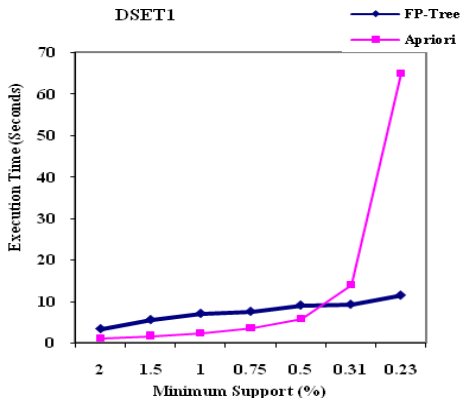

Fig. 5 Performance comparison of Apriori and FP-Growth

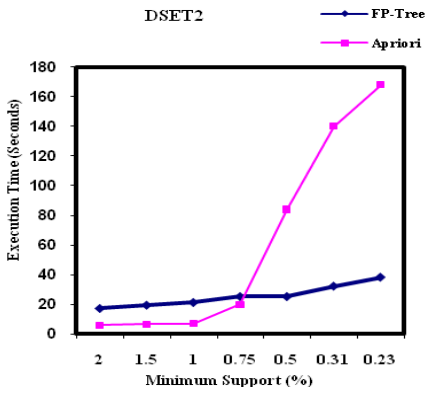

Fig. 6 Performance comparison of Apriori and FP-Growth

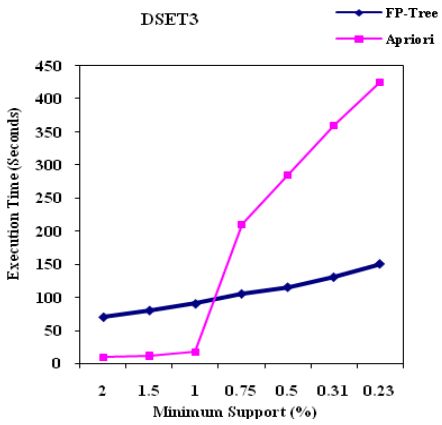

Fig. 7 Performance comparison of Apriori and FP-Growth

The graphs are plotted for the above three datasets where X-axis represents minimum support values and Y-axis represents execution time of the algorithms used. It has been observed from the graphs that as the minimum support values decrease, the execution times algorithms increase exponentially. It is happened due to decrease in the minimum support threshold values make the number of item-sets in the output to be exponentially increased and, it shows the performance of FP-Growth is better than Apriori algorithm for all problem sizes with factor 2 for high minimum support values to very low level support magnitude.

6. Conclusion

In this paper, the association rule mining concept has been explained in a very simple and clear manner with market basket data analysis. Market basket data belongs to large databases category. By using Apriori and FP-Growth

algorithms, it has been shown how frequent patterns dataset are generated with respect to minimum support and minimum confidence count. It has been observed, when the frequency requirements become less the set of association rules grow rapidly and when the frequent item-sets increase, more numbers of association rules are presented to the user, among them many rules found to be redundant. This problem occurs both in transactional as well as spatial data databases and elimination of redundancy of rules are required to be done in a privileged manner. But incase of dense datasets, mining all possible frequent item-sets become less feasible. As in those databases, an exponential number of frequent item-sets are produced; they are let alone to generate rules. A lot of strategies have been proposed for tackling efficiency factor, but always they are found to be successful. Depending upon the association rule mining algorithms the effectiveness of performance is measured. The illustration of market basket data through Apriori and FP-Growth algorithm clearly shows the basics of association rule mining that can be fruitful for everybody to understand about association rule mining. The experimental result depicted how association rule mining can be properly utilized for large database applications as well as both the Apriori and FP-Growth algorithms comparison performance. Our experimental result in the paper justifies, the vital importance of minimum support values for effective performance of different association rule mining algorithms and their acceptability in different problem conditions.

In a nutshell, it can be said that association rules mining actually helps enormously to explore large databases for extracting effective and valuable mining information so that the end users requirement are satisfied with a high level.

References

[1] Victoria Pachon Alvarez, Jacinto Mata Vazquez, An evolutionary algorithm to discover quantitative association rules from huge databases without the need for an a priori discretization, Expert Systems with Applications, Volume 39, Issue 1, January 2012, Pages 585-591.

[2] Chyouhwa Chen, Shi-Jinn Horng, Chin-Pin Huang, Locality sensitive hashing for sampling-based algorithms in association rule mining, Expert Systems with Applications, Volume 38, Issue 10, 15 September 2011, Pages 12388-1239.

[3] Chieh-Ming Wu, Yin-Fu Huang, Generalized association rule mining using an efficient data structure, Expert Systems with Applications, Volume 38, Issue 6, June 2011, Pages 7277-7290.

[4] Hamid Reza Qodmanan, Mahdi Nasiri, Behrouz Minaei-Bidgoli, Multi objective association rule mining with genetic algorithm without specifying minimum support and minimum confidence, Expert Systems with Applications, Volume 38, Issue 1, January 2011, Pages 288-298.

[5] Mourad Ykhlef, A Quantum Swarm Evolutionary Algorithm for mining association rules in large databases, Journal of King Saud University – Computer and Information Sciences (2011) 23, Pages 1–6.

[6] Frank S.C. Tseng, Yen-Hung Kuo, Yueh-Min Huang, Toward boosting distributed association rule mining by data de-clustering, Information Sciences, Volume 180, Issue 22, 15 November 2010, Pages 4263-4289.

[7] Chun-Jung Chu, Vinsent S. Tseng, Tyne Liang, An efficient algorithm for mining high utility itemsets with negative item values in large databases, Applied Mathematics and Computation, Volume 215, Issue 2, 15 September 2009, Pages 767-778.

[8] Anurag Pande, Mohamed Abdel-Aty, Market basket analysis of crash data from large jurisdictions and its potential as a decision support tool Safety Science, Volume 47, Issue 1, January 2009, Pages 145-154.

[9] W a s i l e w s k a , A . - *APRIORI Algorithm*, Lecture Notes, http://www.cs.sunysb.edu/~cse634/lecture_notes/07apriori.pdf, accessed 10.01.2007.

[10] Kotsiantis, S., & Kanellopoulos, R. (2006). Association Rules Mining: A Recent Overview. GESTS International Transactions on Computer Science and Engineering, 32(1), Pages 71-82.

[11] P. Shenoy, J. Haritsa, S. Sudarshan, G. Bhalotia, M. Bawa, and D. Shah. Turbo-charging vertical mining of large databases. In Proc. Of ACM SIGMOD Intl. Conf. on Management of Data, May 2000, Pages 143-160.

[12] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In Proc. Of ACM SIGMOD Intl. Conf. on Management of Data, May 2000, Pages 1-12.

[13] C. Hidber. Online association rule mining. In Proc. Of ACM SIGMOD Intl. Conf. on Management of Data, June 1999, Pages 85-93.

[14] S Lawrence, C Lee Giles. Accessibility of Information on the Web [J].1999, Pages 403:107-109.

[15] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In Proc. Of Intl. conf. on Knowledge Discovery and Data Mining (KDD), August 1997, Pages 432-443.

[16] S. Brin, R. Motwani, J. Ulman, and S. Tsur, Dynamic itemset counting implication rules for market basket data. In Proc. Of ACM SIGMOD Intl. Conf. on Management of Data, May 1997, Pages 207-216.

[17] R. Agrawal and R. Srikant. Fast Algorithms for mining Association rules. In Proc. Of Intl. Conf. on Very Large Databases (VLDB), September 1994 Pages 487-499.

[18] Agrawal, R., Imielinski, T., and Swami, A. N. 1993. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Pages 207-211.

[19] A G. Piatetsky-Shapiro and W. J. Frawley, editors. Knowledge Discovery in Databases. MIT Press, 1991, Pages 1-30.

Biswaranjan Nayak is working as Asst. Profesor in Dept. Of CSE at Trdent Academy of Technology, Bhubaneswar, Odisha, India. He has received his MCA and M.Tech degree in Computer Science from NIELIT, New Delhi. He has more than 18 years of teaching and software développent experience.

Dr. Srinivas Prasad is working as Professor, Dept. Of CSE & Dean (R & D) at GITA, Bhubaneswar,Odisha, India. He has received his PhD. Degree in Comp. Sc. From Utkal University, India. He has received his B.E in Computer Science from A.U, India and M.Tech in Comp. Appl. From ISM Dhanbad, India and M.S. in System Software from BITS Pilani, India. He has more than 20 years of Software development and teaching experience in USA and India.