Finding the Frequent Pattern in a Database
A Study on the Apriori Algorithm

Najlaa AlHuwaishel[1], Maram AlAlwan[2] and Ghada Badr[3]
[1]*Department of Computer Sciences, King Saud University, Riyadh, Saudi Arabia*
[2]*King Abdulaziz City for Science and Technology, Riyadh, Saudi Arabia*
[3]*Department of Computer Sciences, King Saud University, Riyadh, Saudi Arabia*

Keywords: Data Mining, Frequent Pattern, Apriori Algorithm.

Abstract: This paper is a study on the frequent pattern mining using the Apriori algorithm. We will present the concept of data mining in a high level and explain how the frequent pattern is mined. We will talk about the Apriori algorithm and the problem with this algorithm followed by exploration of some algorithms that considered as an update on the Apriori algorithm in order to enhance the efficiency problem that we explained. We will also compare the selected algorithms taking under consideration: 1.Number of database scan. 2.Number of generated candidate lists. 3.Memory usage.

1 INTRODUCTION

In many knowledge fields and industries (or business), it is important to store data and information. There are many technologies supports and continuously develop new efficient trusted methods of storing data. Data can be stored in more than a structure: Database, Data Warehouse, World Wide Web information repository, Spreadsheets etc.

After secure, efficient and reliable Databases; what is next? It is the time that we need to benefit from these data - wither we are working in a scientific or business field. The process of extracting information or knowledge from large set of data is called "Data Mining". One of the definitions of Data Mining is (An essential process where intelligent methods are applied in order to extract data patterns).

2 DATA MINING CONCEPTS

There are different kinds of data pattern that we can mine or extract from a database, depending on the reason of the study of a database:

1. Concept/Class Description: Characterization and Discrimination
Data can be associated with classes or concepts. This kind of pattern interested in the characterization of a group of objects. For example in the owners of a store needs to study classify the tight budget spenders and open budget spenders and each group interests. The output of data characterization process can be presented in various forms: pie charts, bar charts, curves, multidimensional data cubes, and multidimensional tables. The resulting descriptions can also be presented as generalized relations or in rule form (called characteristic rules).

2. Classification and Prediction
Classification is the process of finding a model (or function) that describes and distinguishes data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown. The derived model is based on the analysis of a set of training data (i.e., data objects whose class label is known).

3. Cluster Analysis
Unlike classification and prediction, which analyze class-labeled data objects, clustering analyzes data objects without consulting a known class label. The objects are
Clustered or grouped based on the principle of maximizing the interclass similarity and minimizing the interclass similarity. That is, clusters of objects are formed so that objects within a cluster have high similarity in comparison to one another, but are very dissimilar to objects in other clusters. Each cluster that is formed can be viewed as a class of objects.

4. Evolution Analysis
From its name; Data evolution analysis describes and

AlHuwaishel N., AlAlwan M. and Badr G..
Finding the Frequent Pattern in a Database - A Study on the Apriori Algorithm.
DOI: 10.5220/0005147703880396
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (KDIR-2014), pages 388-396
ISBN: 978-989-758-048-2

models regularities or trends for objects whose behavior changes over time. Although this may include characterization, discrimination, association and correlation analysis, classification, prediction, or clustering of time-related data, distinct features of such an analysis include time-series data analysis, sequence or periodicity pattern matching, and similarity-based data analysis.

5. Outlier Analysis

Which can be the opposite of the Frequent Pattern mining. A database may contain data objects that do not comply with the general behavior or model of the data. These data objects are outliers. Most data mining methods discard outliers as noise or exceptions. However, in some applications such as fraud detection, the rare events can be more interesting than the more regularly occurring ones. The analysis of outlier data is referred to as outlier mining.

6. Mining Frequent Patterns

The Frequent Patterns is the most frequent items that are occurring in a database. There are many kinds of frequent patterns, including item-sets, subsequences, and substructures. A frequent item-set typically refers to a set of items that frequently appear together in a transactional data set, such as milk and bread. A frequently occurring subsequence, such as the pattern that customers tend to purchase first a PC, followed by a digital camera, and then a memory card, is a (frequent) sequential pattern. A substructure can refer to different structural forms, such as graphs, trees, or lattices, which may be combined with item-sets or subsequences. If a substructure occurs frequently, it is called a (frequent) structured pattern. Mining frequent patterns leads to the discovery of interesting associations and correlations within data. A lot of changes can be done relaying of the frequent patterns studies that can benefit the field of interest.

Our concern here in this paper is to explore the Data Mining most famous algorithm (Apriori Algorithm) that is designed to find the most frequent item set in a database and also to find the relation between them which will lead us to better ways in decision making.

3 MINING THE FREQUENT PATTERNS

When we are studying the relationships or the associations between the Frequent Patterns or the Frequent Items there are 2 important concepts we need to understand: Support & Confidence.

An objective measure for association rules of the form $X \Rightarrow Y$ is rule support, representing the percentage of transactions from a transaction database that the given rule satisfies. This is taken to be the probability $P(X \cup Y)$, where $X \cup Y$ indicates that a transaction contains both X and Y , that is, the union of item-sets X and Y .

Another objective measure for association rules is confidence, which assesses the degree of certainty of the detected association. This is taken to be the conditional probability $P(Y|X)$, that is, the probability that a transaction containing X also contains Y. More formally, support and confidence are defined as:

$$\text{support}(X \Rightarrow Y) = P(X \cup Y).$$
$$\text{confidence}(X \Rightarrow Y) = P(Y|X).$$

Example: If we are doing an Association analysis for an electronic store, suppose that the Manager would like to determine which items are frequently purchased together within the same transactions. An example of such a rule, mined from the store transactional database, is

$$\text{buys}(X, \text{“computer”}) \Rightarrow \text{buys}(X, \text{“software”})$$
$$[\text{support} = 1\%, \text{confidence} = 50\%]$$

Where X is a variable representing a customer. A confidence, or certainty, of 50% means that if a customer buys a computer, there is a 50% chance that she/he will buy software as well. A 1% support means that 1% of all of the transactions under analysis showed that computer and software were purchased together.

Dropping the predicate notation, the above rule can be written simply as

$$\text{“computer} \Rightarrow \text{software} \quad [1\%, 50\%]\text{”}$$

This association rule involves a single attribute or predicate (i.e., buys) that repeats. Association rules that contain a single predicate are referred to as *Single-Dimensional Association* Rules.

On the other hand; there is another kind of association which we call the *Multidimensional Association* Rule, which involve more than one attribute in the frequent items relation. Example: For the same pervious electronic store a data mining system may find association rules like

$$\text{age}(X, \text{“20...29”}) \wedge \text{income}(X, \text{“20K...29K”}) \Rightarrow \text{buys}(X, \text{“CD player”})$$
$$[\text{support} = 2\%, \text{confidence} = 60\%]$$

The rule indicates that of the store customers under study, 2% are 20 to 29 years of age with an income of 20,000 to 29,000 and have purchased a CD player.

Also there is a 60% probability that a customer in this age and income group will purchase a CD player.

As the pervious examples, any association rules that are not satisfying the 2 concepts (minimum support threshold and a minimum confidence threshold), are discarded and not considered in the analytical studies.

The association rule mining has received a great deal of attention since its introduction. Today the mining of such rules is still one of the most popular and attracting pattern-discovery methods in Knowledge Discovery and Data mining (KDD). Therefore, the problem of mining association rules can be converted into two sub-issues:

- Identify all the frequent item sets that are set to meet the pre-defined minimum support.
- Find all association rules between the set of all subsets of the frequent item sets, these rules must also meet the minimum support and minimum confidence.

In fact, the performance of association rule mining algorithm focused on the first issue, in which the most algorithms are focused on how to efficiently discover frequent item sets.

4 THE APRIORI ALGORITHM

The Apriori Algorithm considered the basic algorithm for finding a frequent item set in a database. This algorithm was proposed by R. Agrawal and R. Srikant in 1994 for mining frequent item-sets for Boolean association rules.

The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent item-set properties; also the threshold is already known (the minimum support and the minimum confidence).

Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation).

A. How Apriori Algorithm Works

The Apriori Algorithm simply consists of two steps or stages:

1. Find all frequent item-sets:
   a) Get frequent items: Items whose occurrence in database is greater than or equal to the min.support threshold.
   b) Generate candidates from frequent items.
   c) Prune some results (the redundant records or the ones that did not exceed the min.support).
2. Generate Strong rules from frequent item-sets: Rules which satisfy the min.support and min.confidence threshold.

B. Find all frequent item-sets

Apriori Algorithm employs an iterative approach known as a level-wise search, where k-item-sets are used to explore (k + 1)-item-sets. First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted L1. Next, L1 is used to find L2, the set of frequent 2-itemsets, which is used to find L3, and so on, until no more frequent k-item-sets can be found. The finding of each Lk requires one full scan of the database.

Input:

- D, a database of transactions;
- min_sup, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

(1) L_1 = find_frequent_1-itemsets(D);
(2) for ($k = 2; L_{k-1} \neq \phi; k++$) {
(3) C_k = apriori_gen(L_{k-1});
(4) for each transaction $t \in D$ { // scan D for counts
(5) C_t = subset(C_k, t); // get the subsets of t that are candidates
(6) for each candidate $c \in C_t$
(7) c.count++;
(8) }
(9) $L_k = \{c \in C_k | c.count \geq min\_sup\}$
(10) }
(11) return $L = \cup_k L_k$;

To improve the efficiency of the level-wise generation of frequent item-sets, an important property called the Apriori property is used to reduce the search space.

procedure apriori_gen(L_{k-1}:frequent $(k-1)$-itemsets)
(1) for each itemset $l_1 \in L_{k-1}$
(2) for each itemset $l_2 \in L_{k-1}$
(3) if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ then {
(4) $c = l_1 \times l_2$; // join step: generate candidates
(5) if has_infrequent_subset(c, L_{k-1}) then
(6) delete c; // prune step: remove unfruitful candidate
(7) else add c to C_k;
(8) }
(9) return C_k;

procedure has_infrequent_subset(c: candidate k-itemset;
L_{k-1}: frequent $(k-1)$-itemsets); // use prior knowledge
(1) for each $(k-1)$-subset s of c
(2) if $s \notin L_{k-1}$ then
(3) return TRUE;
(4) return FALSE;

Apriori Property: All nonempty subsets of a frequent item-set must also be frequent. The Apriori property is based on the following observation. By definition, if an item-set I does not satisfy the minimum support threshold, min sup, then I is not frequent; that is, P(I) < min_support.
If an item A is added to the item-set I, then the resulting item-set (i.e., I ∪ A) cannot occur more frequently than I. Therefore, I ∪ A is not frequent either; that is, P(I ∪ A) < min_support.

C. Generate Strong rules from frequent item-sets

In Order to complete "Find the Frequent Pattern" process and also to reach to meaningful results that we can benefit from and use; we need to find or generate an association rules or a relationship between these frequent pattern items/item-sets. We can do that by using the following formula that depends on the support & confidence values:

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

5 THE PROBLEM

Apriori algorithm is a classical algorithm that has caused the most discussion. It can effectively carry out the mining association rules. In the Apriori algorithm a set of candidate item-sets Ck is generated from Lk-1. Every item-set in Ck is tested whether it's all k-l subsets constitute a large k-1 item-set or not. If one of k-1 item-sets is not in Lk-1 item-sets, the super item-set of this k-l item-set can be deleted. That is, every time a k item-set is constituted, Apriori must scan all Lk-1 item-sets. Because of many scan large database many times in this way, the identifying processes are the bottleneck of the Apriori algorithm.

Apriori algorithm may need to generate a huge number of candidate generations. Each time when candidate generations are generated, the algorithm needs to judge whether these candidates are frequent item sets. The manipulation with redundancy result in high frequency in querying, so tremendous amounts of resources will be expended whether in time or in space.
As it shows; mostly the problem with the Apriori algorithm is the efficiency. We need to scan all the items in the database multiple times. Also, redundant generation of sub-item-sets during pruning the candidate item-sets. Experiments show that time taken for the database scan is more than the time taken for candidate generation when the database size is large (Tiwari et al., 2009).

We will explore in this paper two of the enhancements and techniques that been added to the Apriori algorithm in order to improve the efficiency.

6 THE STUDY

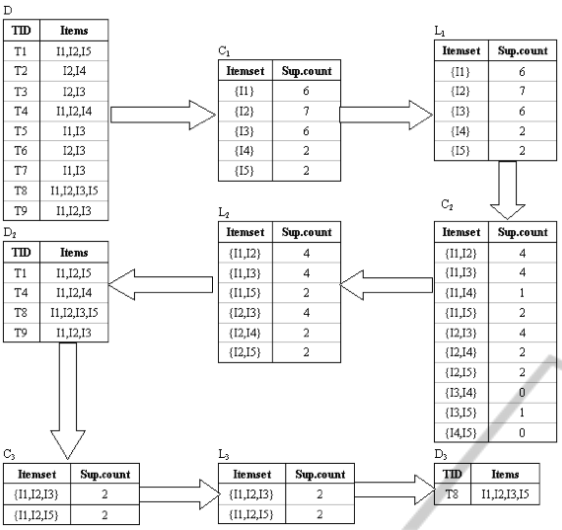
For more than 10 years, a lot of studies and researches were conducted in order to find a solution to enhance the efficiency of the Apriori algorithm. All the studies stated that the Apriori algorithm results are trusted and the only concern was regarding the time that it is taking and the number of redundancy that is generated in the candidate list of frequent item-set list. And with the growth of the databases the problem will become more complex. We explored lots of researches and improvements on the Apriori algorithms, some of them used another structure as a tree, hash or matrixes and some updated the algorithm by combine it with other ways like the FP-tree concept or partition concept. However, this made it more complex and harder to implement. In this paper here we want to highlight some of the papers that studied the Apriori algorithm and updated it in order to improve the efficiency as the following:

1. *The Research of Improved Apriori Algorithm for Mining Association Rules – 2007*

This paper presented an improved Apriori algorithm to increase the efficiency of generating association rules. This algorithm adopts a new method to reduce the redundant generation of sub-item-sets during pruning the candidate item-sets, which can form directly the set of frequent item-sets and eliminate candidates having a subset that is not frequent in the meantime. This algorithm can raise the probability of obtaining information in scanning database and reduce the potential scale of item-sets.

The steps of this new algorithm are:

a) Generate a Candidate list for the frequent items (C1)
b) Get frequent items from the Candidate lest. (L1)
c) Generate candidates from frequent items.(Ck+1 from Lk)
d) Check for infrequent items to be removed or Prune from the candidate list before getting the next L list, and so one.



The update that made here is that there is a candidate list and a check point for the infrequent items to be removed before moving to the next level. The advantages of this approach:

- The size of database is reduced.
- The storage space and computing time are saved.

2. A New Approach to Mine Frequent Itemsets - The Frequent Item-sets Tree (FI-tree)- 2012

Simply; the idea here is to use a tree structure to store the frequent item-sets along with their transaction IDs. A distinct feature of this method is that it has runs fast in different data characteristics. This study showed that this new approach has high performance in various kinds of data, outperforms the previously developed algorithms in different settings, and is highly scalable in mining different databases.

Steps of the proposed approach for mining frequent item-sets from different data characteristics are:

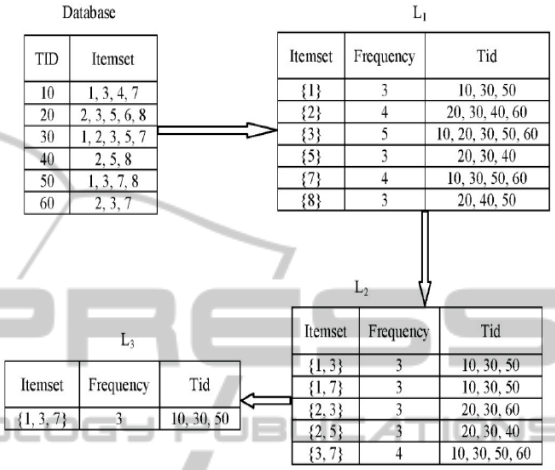
1. Assume that the frequent 1-itemsets and transaction sets, require no more memory that available and also there is space for generating candidate 2-itemsets from frequent 1-itemset. Scan database and find frequent 1-itemsets, at the same time obtain transaction sets, which includes the Item-set.

2. Generate candidate 2-itemsets from frequent 1-itemset only.

3. The candidate 2-itemsets whose node count is lower than min support using their FI-tree data structure, it will be pruned off. Now frequent Item-set tree contains only frequent 2-itemsets at the second level.

4. Consequently, for each frequent 3, 4... n–Item-set, scan the database to approve the consistence of the Item-set.

Also this update is using the same concept of checking and pruning before reaching the final candidate in order to speed up the process and reduce the redundancy as possible.



The advantages of this approach:

- Fast approach as it is using an implicit candidate list by using a frequent item tree structure.

3. A Novel Algorithm for Mining Frequent Item-set From Larg Database - 2009

As the data to be mined is large, the time taken for accessing data is considerable. In this search, a new association rule-mining algorithm which generates the frequent item-sets in a single pass over the database is presented. The algorithm mainly uses two approaches for association rule mining: The Partition approach, where the data is mined in partitions and merges the result, and the Apriori approach that helps to find the frequent sets within each partition. The method proposed in this research is pretty much depending on the (Divide and Conquer) concept.
This partition approach used for finding frequent item-sets in single pass over the database consists of two phases.

Phase 1:
In this phase, the partition algorithm logically divides the database in to a number of non-overlapping partitions. These partitions are considered one at a time and all frequent item-sets for that partition (Li) are generated using the Apriori algorithm. In addition, when taking each partition for calculating the frequent item-sets separately the local minimum support is set to 1. Thus, if there are n partitions, phase I of the algorithm takes n iterations. At the end of phase I, all

the local frequent item-sets of each partition are merged to generate a set of all potential frequent item-sets. In this step, the local frequent item-sets of same lengths from all n-partitions are combined to generate the global candidate item-sets, and also those global candidate item-sets have their combined support associated with it.

Phase 2:

As the above generated global candidate item-sets have least possible frequent item-set of that partition because, during the generation of frequent item-sets using the Apriori algorithm of each partition, the minimum local support was set to 1. So this phase just prune the item-sets from the global candidate item-sets list whose combined support is less than the global minimum support.

Below is the algorithm of modified partition approach:

P = partition_database (T); N = Number of partitions;
// Phase 1
for i = 1 to n do begin
read _in_partition (T_i in P)
L_i = generate all frequent itemsets of Ti using *apriori* method in main memory.
End
// Merge Phase
For ($k = 2$; $L_i^k \neq \Phi$, $i = 1, 2, \ldots\ldots, n$; k^{++}) do begin
$C_k^G = Y_{i=1}{}^n Li^k$
End

// Phase II
$L^G = \Phi$;
for each $c \in C^G$ do begin
if $S(C)_{Tc} \geq \sigma$
$L^G = L^G \cup \{S(C)\}$
End
Answer = L^G

The following notation is used in the remainder of this approach:

Notation	*Meaning*
L^i	*Local Frequent Sets:* Set of Local Frequent Itemsets of Partition i.
C_k^G	*Global Frequent Sets:* Set of global candidate K-Itemsets.
L_i^k	*Local Frequent Sets:* Set of local frequent K-Itemsets in partition i.
L^G	*Global Frequent Itemsets:* Set of global frequent Itemsets.
$S(c)T_c$	*Combined Support* Total support of candidate set c in all partitions.

The advantages of this approach:

- Database is scanned only once.
- It will work most efficient in the case of large databases with smallest minimum support values.

4. *A New Improvement of Apriori Algorithm for Mining Association Rules – 2010*

This enhanced algorithm is based on the user interest and the importance of item-sets. In most existing algorithms, the calculation of support only considered the frequency of item-sets appearances, without taking into account the importance of the item itself. For example, shopping malls sell 10,000 towels one month, but only 300 TV sets, in the traditional association rule-mining algorithm, the TV is likely to be ignored. However, from the view of the total sales amount, TV is far more than a towel, derived from the traditional association rules algorithm is obviously wrong. Therefore, the same consideration of a towel and a TV is obviously unreasonable, need to take into account the frequency of item sets and importance of item sets when calculate the support degree of item set, in this way, association rules will be more scientific and reliable.

If only considering the frequency of item sets, then calculating item set support functions can be defined as:

$$f(X) = \text{numTids}(\text{tids}(X))/ \text{numTids}(\text{tids}(\text{ø}))$$

In order to consider the importance of item sets in calculating support, need introduce a function which compute support degree after quantify the importance of item set, **weight** is a key quantitative indicators of measuring the importance of item set. Because the weight of items is a quantitative indicators that is used to measure the importance of item, when establish a database, can set a individual column to store weight, also could make use of a existing column to measure the weight.

For example, in a database that records the sales records of shopping mall, each item will have a price, suppose that the Px means the price of goods I , Pmax means the price of the most expensive goods, then the weight of goods X can be calculated using the following formula:

$$Wx = Px / Pmax$$

This calculation may be not very appropriate, but it does reflect to some extent the importance of each item. For this algorithm, the most important consideration is that existing algorithms often ignore these items, which have great importance but frequency less than other items. Therefore, such an idea can be used to assign weight to the item in the

item set: increase these support degree of items that have great importance but less frequency. In order to do this, the following formula can be used to assign weight to the item:

$$Wx = (1- \beta) \times numTids\ (tids\ (x)) / \max z\ (numTids\ (tids\ ((z)))$$

Z belongs to item set I, β is a parameter in the range of [0, 1], which control the degree of correlation between the frequency and weight of item. If β = 0, then Wx = 1, that means there is no correlation between the weight and the frequency of items, it is always constant to 1, only considered the frequency when calculate item set and the support of item set. The description and process of algorithm:

$L_1 = init(I);$

for $(k = 2; L_k > k-1; k++)$

$L_k = GenItemSet(L_{k-1}, minsupp);$

$L = U_k L_k;$

R=GenAssociationRule $(L_k, minconf)$;

In which, obtain frequent item set through GenItemSet sub function. Pseudo code as follows:

procedure LargeItemGen $(L_{k-1}, minsupp)$ {

for X and Y is in L_{k-1} {

if first $k-2$ items of X and Y are same and any subset of $X \cup Y$ is in L_{k-1} {

$tids(X \cup Y) = tids(X) \cap tids(Y)$

If $(X \cup Y) = \min(W_x, W_y) / numTids(tids(X \cap Y));$) and f($X \cup Y$)> $minsupp$

insert $X \cup Y$, tids($X \cap Y$) and f($X \cup Y$) into L_k ; }

return L_k ;

}

And obtain association rule from frequent item set through GenAssociationRule sub function, Pseudo code as follows:

procedure GenAssociationRule $(L_k, minconf)$ {

for any $X \in L$ and any nonempty subset Y of X

if $f(X) / f(Y) \geq minconf$

insert $Y \Rightarrow (X - Y)$ into R;

return R; }

Code shown in the above process, first, generate a set of items of user interest as mining object from the database, and then scan the database one time to represent item set with transaction identification number. After generating item-sets, assign weight to element in item set, then use the introduction function of the support of the weight to calculate the item set degree of support to generate frequent item set, finally, obtain association rule from these frequent item set.

So the idea here; the pruning is done prior to the database scan, after the pruning it is re-connected to scan the database, thus reducing the number of scans. The advantages of this approach:

- Consider items which have great importance and less frequency (more accurate results).
- Reduces the number of database scans (the pruning is done prior to the database scan).
- Reduces the storage space.

7 FINAL COMPARISON

Algorithm Name	Algorithm Approaches
Improved Aproiri algorithm	Generate candidate k-itemsets from frequent (k-1)-itemsets. Check if C has infrequent subset(Lk-1), If it returns true it will remove C, Otherwise, scan database D (Scan only the transactions that have size >= K). Compute the frequency of frequent k-itemsets when k-itemsets are generated by (k-1)-itemsets. Find frequent itemsets using an iterative level-wise approach based on candidate generation.
FI-Tree Algorithm	Scan database and find frequent 1-itemsets, at the same time obtain transaction ID. Generate candidate 2-itemsets from frequent 1-itemset only. The candidate 2-itemsets whose node count is lower than min support using their FI-tree, it will be pruned off. (Now frequent Item-set tree contains only frequent 2-itemsets at the second level). Consequently, for each frequent 3, 4... n–Item-set, scan the database to approve the consistence of the Item-set.
Partition algorithm	Divides the database in to n partitions. Calculate frequent Item-Set using Aproiri (Set min-sup to 1) in each partition. Merges all the local frequent item-sets of the same lengths in each partition to generate a global frequent item-sets Prune the global candidate item-sets whose combined support is less than the global min-sup.
Weight algorithm	Generate a set of items of user interest from the database. Scan the database one time to represent item set with Tids. Assign weight to each element in item set. Use the function to calculate the support degree of items to generate frequent item set. Obtain association rule from these frequent item set.

Algorithm Name	Database Scan	Memory Usage	Candidate Generation
Improved Aproiri algorithm	Same number of database scans in aproiri but less time in each scan.	Reduces the memory space (scan only the transactions with items>=K).	Same number of candidate generation in aproiri.
FI-Tree Algorithm	Less than aproiri, it is checking and Pruning before reaching the final candidate.	Assume that it will not require any more memory	Less number of candidate lists generated than aproiri stored in tree.
Partition algorithm	One scan	Reduces the memory space (Each partition can adapting better in memory).	More than aproiri
Weight algorithm	Less than aproiri, where it considers only items with Support degree>= min-supp after the first scan)	Reduces the memory space	Less number of candidate lists generated than aproiri

8 EXAMPLES OF APRIORI APPLICATIONS

Apriori algorithm concept is used in real life in many fields with different level of complexity, from DNA researches to a simple supermarket store financial analysis tools. As an Examples:

1. Application of the Apriori algorithm for adverse drug reaction detection - 2009 (US National Library of Medicine/National Institutes of Health)

A research was intend to check for the detection of adverse drug reactions (ADR) in health care data using Apriori Algorithm. The Apriori algorithm is used to perform association analysis on the characteristics of patients, the drugs they are taking, their primary diagnosis, co-morbid conditions, and the ADRs or adverse events (AE) they experience. This analysis produces association rules that indicate what combinations of medications and patient characteristics lead to ADRs. A simple data set is used to demonstrate the feasibility and effectiveness of the algorithm.

2. Apriori Application To Pattern Profile Creditor Relationships With Credit Ceiling In Rural Bank - 2012 (Diponegoro University / Indonesia)

Maintaining the customer to remain loyal and acquiring new clients are the successful capital for financial institutions such as the BPR. One marketing strategy is obtained from a Database, which is a combination of hidden patterns of relationships between items on the profile of creditors with credit ceiling. Pattern rules creditor profile relationship with the ceiling is presented in the model Apriori application. Item profiles creditors and the credit ceilings that appear simultaneously on each transaction are important in finding new marketing strategies such as target new creditors. The results of the application may present Apriori pattern profile creditor relationship with the credit ceiling with graphs to provide a benchmark in providing the credit limit on case that has never happened.

9 CONCLUSIONS

We have explored the Apriori Algorithm concept in the field of minding the frequent item sets from a database along with some enhancements and updates on the classical Apriori algorithm that we considered as the best simple improvements.

As a result of our study and after the comparison that we made between the algorithms that we explored, we found that no need to complex the process as the Apriori algorithm is simple enough to be trusted and rely on. However the problem was the efficiency itself, which was taking some time to scan the database more than once to get the candidate list and also the redundancy problem, which is also affecting the process. So if the Apriori algorithm will be used in an application with a dedicated database it is better and more efficient to use the classic Apriori algorithm. On the other hand and with any special conditions or under any special circumstances we should take under consideration and choose a proper updated Apriori to work most efficient.

REFERENCES

Jiawei Han and Micheline Kamber, "Data Mining: Concepts and Techniques, Second Edition," 2006.

Akhilesh Tiwari, Rajendra K. Gupta and Dev Prakash Agrawal, "A Novel Algorithm for Mining Frequent Item-set From Larg Database", In International Journal

of Information Technology and Knowledge Management, Volume 2, No. 2, July-December 2009, pp. 223-229.

Libing Wu, KuiGong, Fuliang Guo, XiaohuaGe, Yilei Shan, "Research on Improving Apriori Algorithm Based on Interested Table", 2010 IEEE, pp.422-426.

Guofeng Wang, Xiu Yu, Dongbiao Peng, Yinhu Cui, Qiming Li, " Research of Data Mining Based on Apriori algorithm in Cutting Database", 2010 IEEE

Du Ping, Gao Yongping, " A New Improvement of Apriori Algorithm for Mining Association Rules", In International Conference on Computer Application and System Modeling (ICCASM), 2010, pp.529-532

Patel Tushar S. and Amin Kiran R., "A New Approach to Mine Frequent Itemsets", In ISCA Journal of Engineering Science, Vol. 1, July 2012, pp. 14-18.

Sheng Chai, Jia Yang, Yang Cheng," The Research of Improved Apriori Algorithm for Mining Association Rules ",2007 IEEE.

Anurag Choubey, Ravindra Patel, J.L. Rana," A Survey of Efficient Algorithms and New Approach for Fast Discovery of Frequent Itemset for Association Rule Mining (DFIARM) ", In International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-1, Issue-2, May 2011, pp.62-67.

Tang Junfang, " An Improved Algorithm of Apriori Based on Transaction Compression", In 2nd International Conference on Control, Instrumentation and Automation (ICCIA), 2011, pp.356-358.

Rui Chang, Zhiyi Liu, " An Improved Apriori Algorithm", In International Conference on Electronics and Optoelectronics (ICEOE),2011, pp.476-478.

Preetham Kumar, Ananthanarayana V S, "Parallel Method for Discovering Frequent Itemsets Using Weighted Tree Approach", In International Conference on Computer Engineering and Technology,2009, pp.124-128.

Li Tu, Ling Chen, Shan Zhang, "Efficient Algorithms with Time Fading Model for Mining Frequent Items over Data Stream", In International Conference on Industrial and Information Systems, 2009, pp.403-408.

R.M. Karp, S. Shenker, and C.H. Papadimitriou, "A simple algorithm for finding frequent elements in streams and bags", presented at ACM Transactions on Database Systems, Vol. 28, No. 1, March 2003, pp.51-55.

Pramod S, Vyas O.P, "Survey on Frequent Item set Mining Algorithms," presented at International Journal of Computer Applications, Volume 1 – No. 15 pp.86-91.