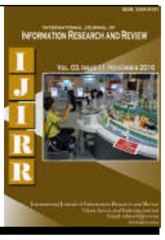
IJIRR

International Journal of Information Research and Review
Vol. 03, Issue, 11, pp. 3193-3216, November, 2016



Research Article

AN OVERVIEW OF MODIFIED APROIRI ALGORITHM ON FREQUENT TRAJECTORY SYSTEM OF AN OBJECT: EMERGING OPPORTUNITIES OF GAME THEORY APPLICATION

***Nihar Ranjan Hota, Ipsit Ranjan Joshi Manas Ranjan Moharana and Jyostna Rani Rout**

Assistant Professor in Dept. of Computer Science& Engineering, Einstein Academy of Technology & Management (EATM), Bhubaneswar (India) Approved by AICTE &Affiliated to Biju Patnaik University and Technology (BPUT) Rourkela, Odisha, India

ARTICLE INFO

Article History:

Received 18th August, 2016
Received in revised form
22nd September, 2016
Accepted 14th October, 2016
Published online November, 30th 2016

Keywords:

Data Mining,
Modified Apriori Algorithm,
Sequential pattern Mining Method,
Frequent Itemset Generation,
Game Theory.

ABSTRACT

Frequent pattern mining has been an emerging and active field in data mining research for over a decade. Abundant literature has been emerged from this research and tremendous progress has been made in numerous research frontiers. This article, provide an application of the modified Apriori algorithm in coordinate sets of trajectories to find the frequent trajectory coordinates. In this algorithm additional steps are added to prune the coordinate setsgenerated so that to reduce the unnecessary search time and space.This sequential pattern mining method is quite simple in naturebut complex to implement. This paper explains the basics of dataorigination, database structure to hold the coordinate datasets andthe implementation of the algorithm with the object orientedprogramming language by an illustration. It can be applied tointeresting game theory domains to find the frequent trajectory of anobject shot by a player which follows a trajectory path.

INTRODUCTION

Apriori is a classic algorithm for frequent itemset mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis.

1.1Earlier Work

Association mining systems that have been developed withclassification purposes in mind are sometimes dubbedclassification rule mining. Some of these techniques can beadapted to our needs. Take, for instance, the approach proposed, if $i_j$is the item whose absence or presence is to be predicted,the technique can be used to generate all rules that havethe form $r^{(a)} \Rightarrow I_j$, where $r^{(a)} \subseteq (I \backslash \{i_j\})$ and I_j is thebinary class label (i_j= present or i_j= absent). For a givenitemset s, the technique identifies among the rules withantecedents subsumed by s those that have the highestprecedence according to the reliability of the rules—thisreliability is assessed based on the rules' confidence andsupport values. The rule is then used for the prediction ofi_j. The method suffers from three shortcomings. First, it isclearly not suitable in domains with many distinct itemsi_j. Second, the consequent is predicted based on the"testimony" of a single rule, ignoring the simple fact thatrules with the same antecedent can imply different consequents—a method to combine these rules is needed. Third, the system may be sensitive to the subjective user specifiedsupport and confidence thresholds. Some of these weaknesses are alleviated in, wherea missing item is predicted in four steps. First, they use aso-called partitioned-ARM to generate a set of associationrules (a ruleset). The next step prunes the ruleset (e.g., byremoving redundant rules).

****Corresponding author: Nihar Ranjan Hota,***
Assistant Professor in Dept. of Computer Science& Engineering, Einstein Academy of Technology & Management (EATM), Bhubaneswar (India) Approved by AICTE &Affiliated to Biju Patnaik University and Technology (BPUT) Rourkela, Odisha, India.

From these, rules with thesmallest distance from the observed incomplete shoppingcart are selected. Finally, the items predicted by theserules are weighed by the rules' antecedents' similarity tothe shopping cart. The approach in pursues a Dempster-Shafer (DS) belief theoretic approach that accommodates general dataimperfections. To reduce the computational burden, Hewawasamet al. employ a data structure called a belief itemsettree. Here, too, rule generation is followed by a pruningalgorithm that removes redundant rules. In order to predictthe missing item, the technique selects a "matching" ruleset—a rule is included in the matching ruleset if theincoming itemset is contained in rule antecedent. If no rulessatisfy this condition, then, from those rules that havenonempty intersection with the item sets, rules whoseantecedents are "closer" to s according to a given distancecriterion (and a user-defined distance threshold) are picked.Confidence of the rule, its "entropy," and the length of itsantecedent are used to assign DS theoretic parameters to therule. Finally, the evidence contained in each rule belongingto the matching ruleset is combined or "pooled" via aDS theoretic fusion technique.

In principle, at least, we could adopt any of the abovemethodologies; but the trouble is that they were alldesigned primarily for the classification task and not forshopping cart completion. Specifically, the number of timessuch classifiers have to be invoked would be equal to thenumber of all distinct items in the database (i.e., n) minusthe number of those already present in the shopping cart.This is why we sought to develop a predictor that wouldpredict all items in a computationally tractable manner. Another aspect of these approaches is the enormousamount of effort/cost it takes to obtain a tangible andmeaningful set of rules. The root of the problem lies in the apriori-like algorithms used to generate frequent itemsets andthe corresponding association rules—the costs becomeprohibitive when the database is large and complicated.Here, the size and difficulty are determined by fourparameters: number of transactions, number of distinctitems, average transaction length, and the minimum supportthreshold. For example, the problem can become intractableif the number of frequent items is large; and whether an itemis frequent or not is affected by the minimum supportthreshold. It is well known that apriori-based algorithmssuffer from performance degradation in large-scale problemsdue to combinatorial explosion and repeated passes through the database.

1.2 Related Work

As far as we know, the Apriori algorithm has not been studied in any significant way for efficient hardware implementation. However, research in hardware implementations of related datamining algorithms has been done. In the k-means clustering algorithm in implemented as an example of a special reconfigurable fabric in the form of a cellular array connected to a hostprocessor. K-means clustering is a datamining strategy that groups together elementsbased on a distance measure. The distance can be an actual measure of Euclideandistance or can be mapped from any manner of other data types. Each item in aset is randomly assigned to a cluster, the centres of the clusters are computed, andthen elements are added and removed from clusters to more efficiently move themcloser to the centres of the clusters. This is related to the Apriori algorithm as both are dependent on efficient set additions and computations performed on all elementsof those sets, but adds the distance computation and significantly changes how the sets are built up. Besides differing in the overall algorithm, the structure of thecomputation is also significantly different, as the system requires the use of globalmemory, in which each unit's personal memory is accessible by the host controller.By avoiding global connections that violate the principles of systolic design, we canincrease overall system clock frequency and ease routing problems. In a system is implemented which attempts to mediate the high cost of datatransfers for large data sets. Common databases can easily extend beyond the capacity of the physical memory, and slow tertiary storage, e.g., hard drives, are broughtinto the datapath. This paper proposes the integration of simple computational structure for datamining onto the hard drive controller itself. The datamining proposed bythe paper is not Apriori, but rather the problem of exact and inexact string matching,a much more computationally regular problem compared to the Apriori algorithm.However, the work is useful, and will become more so as FPGA performance scales up and significantly exceeds the data supply capabilities of hierarchical memory systems.We base our comparisons of hardware performance versus an efficient software implementation using a tree approach as we are unaware of any comparable hardware implementations of the Apriori algorithm. Extensive research exists onparallelizing correlation algorithms, but we focus on single machine performance.

2.Classification and Prediction

2.1 Classification

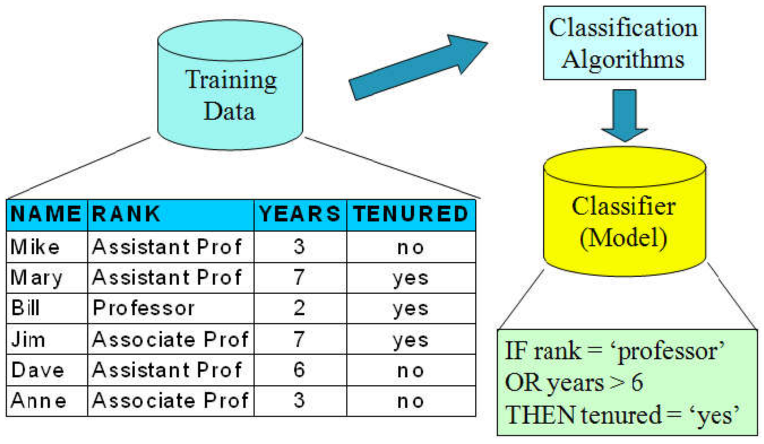
Classification is the process of finding a model (or function) that describes and distinguishes data classes or concepts. The model is derived based on the analysis of a set of training data (i.e., data objects for which the class labels are known). The model is used to predict the class label of objects for which the class label is unknown. *"How is the derived model presented?"* The derived model may be represented in various forms, such as *classification rules* (i.e., *IF-THEN rules*), *decision trees*, *mathematical formulae*, or *neural networks*. A decision tree is a flowchart-like tree structure, where each node denotes a test on an attribute value, each branch represents an outcome of the test, and tree leaves represent classes or class distributions. Decision trees can easilybe converted to classification rules. A neural network, when used for classification, is typically a collection of neuron-like processing units with weighted connections between the units. There are many other methods for constructing classification models, such as naïve Bayesian classification, support vector machines, and *k*-nearest-neighbor classification. Whereas classification predicts categorical (discrete, unordered) labels, regression models continuous-valued functions. That is, regression is used to predict missing or unavailable *numerical data values* rather than (discrete) class labels. The term *prediction* refers to both numeric prediction and class label prediction. Regression analysis is a statistical methodology that is most often used for numeric prediction, although other methods exist as well.

Regression also encompasses the identification of distribution *trends* based on the available data.

2.2 Classification Process

2.2.1 Model construction

It will describe a set of predetermined classes. Each topple/sample is assumed to belong to a predefined class, as determined by the class label attribute. The set of topple used for model construction: training set. The model is represented as classification rules, decision trees, or mathematical formulae.

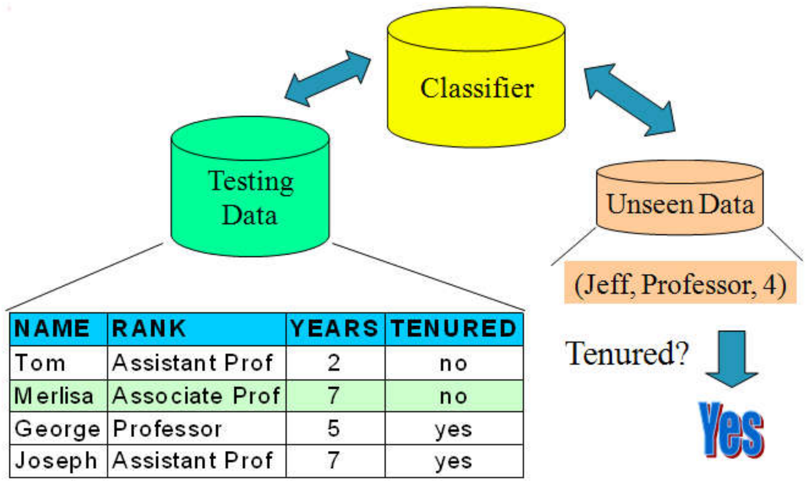


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Figure 2.1. Model construction

2.2.2 Model usage

Estimate accuracy of the model. The known label of test sample is compared with the classified result from the model. Accuracy rate is the percentage of test set samples that are correctly classified by the model. Test set is independent of training set, otherwise over-fitting will occur.



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Figure 2.2: Model usage

2.3 Issues regarding classification and prediction

2.3.1 Data cleaning

Every dataset contains some errors, and every analyst experiences a rite of passage in wasting days drawing wrong conclusions because the errors have not been first rooted out. Up to half of the time needed for analysis is typically spent in "cleaning" the data. This time is also, typically, underestimated. Often, once a clean dataset is achieved, the analysis itself is quite straightforward.Preprocess data in order to reduce noise and handle missing values.

2.3.2Relevance analysis (feature selection)

We rst present a classic notion of feature relevance and illustratewhy it alone cannot handle feature redundancy, and then provide our formal de nition of featureredundancy which paves the way for efficient elimination of redundant features. Remove the irrelevant or redundant attributes.

2.3.3 Data transformation

Data transformation operations, such as normalization and aggregation, are additional data pre-processing procedures that would contribute toward the success of the mining process. It is the processes of representing the collected data in an accurate and compact way without losing any information, it also involves getting a information from collected data.Ex: Display the data as a graph and get the mean, median, mode etc.

2.4 Classification by Decision Tree Induction

A decision tree is a classi er expressed as a recursive partition of the instance space. The decision tree consists of nodes that form a rooted tree,meaning it is a directed tree with a node called a “root” that has no incoming edges. All other nodes have exactly one incoming edge. A nodewith outgoing edges is referred to as an “internal” or “test” node. All othernodes are called “leaves” (also known as “terminal” or “decision” nodes).In the decision tree, each internal node splits the instance space into two ormore sub-spaces according to a certain discrete function of the input attribute values. In the simplest and most frequent case, each test considersa single attribute, such that the instance space is partitioned according tothe attributes value. In the case of numeric attributes, the condition refers.

Each leaf is assigned to one class representing the most appropriate target value. Alternatively, the leaf may hold a probability vector (a nityvector) indicating the probability of the target attribute having a certainvalue. Figure 1.4 describes another example of a decision tree that reasonswhether or not a potential customer will respond to a direct mailing. Internal nodes are represented as circles, whereas leaves are denoted as triangles.Two or more branches may grow from each internal node (i.e. not a leaf).Each node corresponds with a certain characteristic and the branches correspond with a range of values. These ranges of values must give a partitionof the set of values of the given characteristic.

Instances are classi ed by navigating them from the root of the treedown to a leaf, according to the outcome of the tests along the path.Speci cally, we start with a root of a tree; we consider the characteristic that corresponds to a root; and we de ne to which branch the observedvalue of the given characteristic corresponds. Then we consider the nodein which the given branch appears. We repeat the same operations for thisnode etc., until we reach a leaf.

Note that this decision tree incorporates both nominal and numericattributes. Given this classi er, the analyst can predict the response of apotential customer (by sorting it down the tree), and understand the behavioral characteristics of the entire potential customerpopulation regardingdirect mailing. Each node is labeled with the attribute it tests, and itsbranches are labeled with its corresponding values.

In case of numeric attributes, decision trees can be geometrically interpreted as a collection of hyper planes, each orthogonal to one of the axes.

2.5 Bayesian Classification

If one or several attributes or features $a_i \in A$ occur together in more than one itemset (data sample, target data) assigned the topic T , then output a rule

$$Rule_\lambda : a_i \wedge a_j ... \wedge a_m \Rightarrow T$$ or

$$Rule_\mu = \prod_{t=1}^{v} P(a_t \mid X) > \tau_{thres} \rightarrow T_\mu$$

Examples. Naïve Bayes Classifiers

Best for classifying texts, documents, ...

Major drawback: unrealistic independent assumption among individual items.

Basic issue here: Do we accept a document d in class C? If we do, what is the penalty for misclassification?
For a good mail classifier, a junk mail should be assigned "junk" label with a very high probability. The cost of doing this to a good mail is very high.

The probability that a document d_i belongs to topic C_j is computed by Bayes' rule

$$P(C_j \mid d_i) = \frac{P(d_i \mid C_j)P(C_j)}{P(d_i)} \tag{1}$$

Define priori odds on C_j as

$$O(C_j) = \frac{P(C_j)}{1 - P(C_j)} \qquad (2)$$

Then Bayes' equation gives us the posterior odds

$$O(C_j | d_i) = O(C_j)\frac{P(d_i | C_j)}{P(d_i | \square C_j)} = O(C_j)L(d_i | C_j) \qquad (3)$$

Where $L(d_i | C_j)$ is the likelihood ratio. This is one way we could use the classifier to yield posterior estimate of a document. Another way would be to go back to (1). Here

$$P\left(C_j | d_i\right) = \frac{P\left(d_i | C_j\right)P\left(C_j\right)}{P\left(d_i\right)}$$

With

$$P\left(C_j\right) = \frac{n_d\left(C_j\right)}{|D|} \qquad (4)$$

Where $|D|$ is the total volume of the documents in the database, and $n_d\left(C_j\right)$ is the number of documents in class C_j.

Cluster Analysis

Basic Concepts and Methods

Imagine that you are the Director of Customer Relationships at *AllElectronics*, and you have five managers working for you. You would like to organize all the company's customers into five groups so that each group can be assigned to a different manager. Strategically, you would like that the customers in each group are as similar as possible. Moreover, two given customers having very different business patterns should not be placed in the same group. Your intention behind this business strategy is to develop customer relationship campaigns that specifically target each group, based on common features shared by the customers per group. What kind of data mining techniques can help you to accomplish this task?

3.1 What Is Cluster Analysis?

Cluster analysis or simply clustering is the process of partitioning a set of data objects (or observations) into subsets. Each subset is a cluster, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters. The set of clusters resulting from a cluster analysis can be referred to as a clustering. In this context, different clustering methods may generate different clustering's on the same data set. The partitioning is not performed by humans, but by the clustering algorithm. Hence, clustering is useful in that it can lead to the discovery of previously unknown groups within the data. Cluster analysis has been widely used in many applications such as business intelligence, image pattern recognition, Web search, biology, and security. In business intelligence, clustering can be used to organize a large number of customers into groups, where customers within a group share strong similar characteristics. This facilitates the development of business strategies for enhanced customer relationship management. Moreover, consider a consultant company with a large number of projects. To improve project management, clustering can be applied to partition projects into categories based on similarity so that project auditing and diagnosis (to improve project delivery and outcomes) can be conducted effectively.

In image recognition, clustering can be used to discover clusters or "subclasses" in handwritten character recognition systems. Suppose we have a data set of handwritten digits, where each digit is labeled as either 1, 2, 3, and so on. Note that there can be a large variance in the way in which people write the same digit. Take the number 2, for example. Some people may write it with a small circle at the left bottom part, while some others may not. We can use clustering to determine subclasses for "2," each of which represents a variation on the way in which 2 can be written. Using multiple models based on the subclasses can improve overall recognition accuracy. Clustering has also found many applications in Web search. For example, a keyword search may often return a very large number of hits (i.e., pages relevant to the search) due to the extremely large number of web pages. Clustering can be used to organize the search cluster analysis or simply clustering is the process of partitioning a set of data objects (or observations) into subsets. Each subset is a cluster, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters. The set of clusters resulting from a cluster analysis can be referred to as a clustering. In this context, different clustering methods may generate different clustering's on the same data set. The partitioning is not performed by humans, but by the clustering algorithm. Hence, clustering is useful in that it can lead to the discovery of previously unknown groups within the data. Cluster analysis has been widely used in many applications such as business intelligence, image pattern recognition, Web search, biology, and security. In business intelligence, clustering can be used to organize a large number of customers into groups, where customers within a group share strong similar characteristics. This facilitates the development of business strategies for enhanced customer relationship management.

Moreover, consider a consultant company with a large number of projects. To improve project management, clustering can be applied to partition projects into categories based on similarity so that project auditing and diagnosis (to improve project delivery and outcomes) can be conducted effectively. In image recognition, clustering can be used to discover clusters or "subclasses" in handwritten character recognition systems. Suppose we have a data set of handwritten digits, where each digit is labeled as either 1, 2, 3, and so on. Note that there can be a large variance in the way in which people write the same digit. Take the number 2, for example. Some people may write it with a small circle at the left bottom part, while some others may not. We can use clustering to determine subclasses for "2," each of which represents a variation on the way in which 2 can be written. Using multiple models based on the subclasses can improve overall recognition accuracy.

3.2 Requirements for Cluster Analysis

Clustering is a challenging research field. In this section, you will learn about the requirements for clustering as a data mining tool, as well as aspects that can be used for comparing clustering methods.

The following are typical requirements of clustering in data mining. Scalability: Many clustering algorithms work well on small data sets containing fewer than several hundred data objects; however, a large database may contain millions or even billions of objects, particularly in Web search scenarios. Clustering on only a sample of a given large data set may lead to biased results. Therefore, highly scalable clustering algorithms are needed.

- Ability to deal with different types of attributes: Many algorithms are designed to cluster numeric (interval-based) data. However, applications may require clustering other data types, such as binary, nominal (categorical), and ordinal data, or mixtures of these data types. Recently, more and more applications need clustering techniques for complex data types such as graphs, sequences, images, and documents.
- Discovery of clusters with arbitrary shape: Many clustering algorithms determine clusters based on Euclidean or Manhattan distance measures (Chapter 2). Algorithms based on such distance measures tend to find spherical clusters with similar size and density. However, a cluster could be of any shape. Consider sensors, for example, which are often deployed for environment surveillance. Cluster analysis on sensor readings can detect interesting phenomena. We may want to use clustering to find the frontier of a running forest fire, which is often not spherical. It is important to develop algorithms that can detect clusters of arbitrary shape.
- Requirements for domain knowledge to determine input parameters: Many clustering algorithms require users to provide domain knowledge in the form of input parameters such as the desired number of clusters. Consequently, the clustering results may be sensitive to such parameters. Parameters are often hard to determine, especially for high-dimensionality data sets and where users have yet to grasp a deep understanding of their data. Requiring the specification of domain knowledge not only burdens users, but also makes the quality of clustering difficult to control.
- Ability to deal with noisy data: Most real-world data sets contain outliers and/or missing, unknown, or erroneous data. Sensor readings, for example, are often noisy—some readings may be inaccurate due to the sensing mechanisms, and some readings may be erroneous due to interferences from surrounding transient objects. Clustering algorithms can be sensitive to such noise and may produce poor-quality clusters. Therefore, we need clustering methods that are robust to noise.
- Incremental clustering and insensitivity to input order: In many applications, incremental updates (representing newer data) may arrive at any time. Some clustering algorithms cannot incorporate incremental updates into existing clustering structures and, instead, have to recompute a new clustering from scratch. Clustering algorithms may also be sensitive to the input data order. That is, given a set of data objects, clustering algorithms may return dramatically different clustering's depending on the order in which the objects are presented. Incremental clustering algorithms and algorithms that are insensitive to the input order are needed.
- Capability of clustering high-dimensionality data: A data set can contain numerous dimensions or attributes. When clustering documents, for example, each keyword can be regarded as a dimension, and there are often thousands of keywords. Most clustering algorithms are good at handling low-dimensional data such as data sets involving only two or three dimensions. Finding clusters of data objects in a high-dimensional space is challenging, especially considering that such data can be very sparse and highly skewed.
- Scalability: Many clustering algorithms work well on small data sets containing fewer than several hundred data objects; however, a large database may contain millions or even billions of objects, particularly in Web search scenarios. Clustering on only a sample of a given large data set may lead to biased results. Therefore, highly scalable clustering algorithms are needed.
- Ability to deal with different types of attributes: Many algorithms are designed to cluster numeric (interval-based) data. However, applications may require clustering other data types, such as binary, nominal (categorical), and ordinal data, or mixtures of these data types. Recently, more and more applications need clustering techniques for complex data types such as graphs, sequences, images, and documents.
- Discovery of clusters with arbitrary shape: Many clustering algorithms determine clusters based on Euclidean or Manhattan distance measures (Chapter 2). Algorithms based on such distance measures tend to find spherical clusters with similar size and density. However, a cluster could be of any shape. Consider sensors, for example, which are often deployed for environment surveillance. Cluster analysis on sensor readings can detect interesting phenomena. We may want to use clustering to find the frontier of a running forest fire, which is often not spherical. It is important to develop algorithms that can detect clusters of arbitrary shape.

- Requirements for domain knowledge to determine input parameters: Many clustering algorithms require users to provide domain knowledge in the form of input parameters such as the desired number of clusters. Consequently, the clustering results may be sensitive to such parameters. Parameters are often hard to determine, especially for high-dimensionality data sets and where users have yet to grasp a deep understanding of their data. Requiring the specification of domain knowledge not only burdens users, but also makes the quality of clustering difficult to control.
- Ability to deal with noisy data: Most real-world data sets contain outliers and/or missing, unknown, or erroneous data. Sensor readings, for example, are often noisy—some readings may be inaccurate due to the sensing mechanisms, and some readings may be erroneous due to interferences from surrounding transient objects. Clustering algorithms can be sensitive to such noise and may produce poor-quality clusters. Therefore, we need clustering methods that are robust to noise.
- Incremental clustering and insensitivity to input order: In many applications, incremental updates (representing newer data) may arrive at any time. Some clustering algorithms cannot incorporate incremental updates into existing clustering structures and, instead, have to recompute a new clustering from scratch. Clustering algorithms may also be sensitive to the input data order. That is, given a set of data objects, clustering algorithms may return dramatically different clusterings depending on the order in which the objects are presented. Incremental clustering algorithms and algorithms that are insensitive to the input order are needed.
- Capability of clustering high-dimensionality data: A data set can contain numerous dimensions or attributes. When clustering documents, for example, each keyword can be regarded as a dimension, and there are often thousands of keywords. Most clustering algorithms are good at handling low-dimensional data such as data sets involving only two or three dimensions. Finding clusters of data objects in a high-dimensional space is challenging, especially considering that such data can be very sparse and highly skewed.

3.3 Partitioning Methods

The simplest and most fundamental version of cluster analysis is partitioning, which organizes the objects of a set into several exclusive groups or clusters. To keep the problem specification concise, we can assume that the number of clusters is given as background knowledge. This parameter is the starting point for partitioning methods. Formally, given a data set, D, of n objects, and k, the number of clusters to form, a partitioning algorithm organizes the objects into k partitions ($k \leq n$), where each partition represents a cluster. The clusters are formed to optimize an objective partitioning criterion, such as a dissimilarity function based on distance, so that the objects within a cluster are "similar" to one another and "dissimilar" to objects in other clusters in terms of the data set attributes.

3.3.1 k-Means: A Centroid-Based Technique

Suppose a data set, D, contains n objects in Euclidean space. Partitioning methods distribute the objects in D into k clusters, $C1$, …, Ck, that is, $Ci \subset D$ and $Ci \cap Cj =$ for ($1 \leq i, j \leq k$). An objective function is used to assess the partitioning quality so that objects within a cluster are similar to one another but dissimilar to objects in other clusters. This is, the objective function aims for high intracluster similarity and low intercluster similarity. A centroid-based partitioning technique uses the *centroid* of a cluster, Ci, to represent that cluster. Conceptually, the centroid of a cluster is its center point. The centroid can be defined in various ways such as by the mean or medoid of the objects (or points) assigned to the cluster. The difference between an object $p \in Ci$ and ci, the representative of the cluster, is measured by *dist* (p, ci), where *dist* (x, y) is the Euclidean distance between two points x and y. The quality of cluster Ci can be measured by the within-cluster variation, which is the sum of *squared error* between all objects in Ci and the centroid ci, defined as where E is the sum of the squared error for all objects in the data set; p is the point in space representing a given object; and ci is the centroid of cluster Ci (both p and ci are multidimensional). In other words, for each object in each cluster, the distance from the object to its cluster center is squared, and the distances are summed. This objective function tries to make the resulting k clusters as compact and as separate as possible. Optimizing the within-cluster variation is computationally challenging. In the worst case, we would have to enumerate a number of possible partitioning that are exponential to the number of clusters, and check the within-cluster variation values. It has been shown that the problem is NP-hard in general Euclidean space even for two clusters (i.e., $k = 2$). Moreover, the problem is NP-hard for a general number of clusters k even in the 2-D Euclidean space. If the number of clusters k and the dimensionality of the space d are fixed, the problem can be solved in time $O(ndk + 1 \log n)$, where n is the number of objects. To overcome the prohibitive computational cost for the exact solution, greedy approaches are often used in practice. A prime example is the k-means algorithm, which is simple and commonly used. *"How does the* k*-means algorithm work?"* The k-means algorithm defines the centroid of a cluster as the mean value of the points within the cluster. It proceeds as follows. First, it randomly selects k of the objects in D, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the Euclidean distance between the object and the cluster mean. The k-means algorithm then iteratively improves the within-cluster variation. For each cluster, it computes the new mean using the objects assigned to the cluster in the previous iteration. All the objects are then reassigned using the updated means as the new cluster centers. The iterations continue until the assignment is stable, that is, the clusters formed in the current round are the same as those formed in the previous round.

3.4 Hierarchical Methods

While partitioning methods meet the basic clustering requirement of organizing a set of objects into a number of exclusive groups, in some situations we may want to partition our data into groups at different levels such as in a hierarchy. A hierarchical clustering method works by grouping data objects into a hierarchy or "tree" of clusters.Representing data objects in the form of a hierarchy is useful for data summarization and visualization. For example, as the manager of human resources at *AllElectronics*, you may organize your employees into major groups such as executives, managers, and staff. You can further partition these groups into smaller subgroups. For instance, the general group of staff can be further divided into subgroups of senior officers, officers, and trainees. All these groups form a hierarchy. We can easily summarize or characterize the data that are organized into a hierarchy, which can be used to find, say, the average salary of managers and of officers. Consider handwritten character recognition as another example. A set of handwriting samples may be first partitioned into general groups where each group corresponds to a unique character. Some groups can be further partitioned into subgroups since a character may be written in multiple substantially different ways. If necessary, the hierarchical partitioning can be continued recursively until a desired granularity is reached. In the previous examples, although we partitioned the data hierarchically, we did not assume that the data have a hierarchical structure (e.g., managers are at the same level in our *AllElectronics* hierarchy as staff). Our use of a hierarchy here is just to summarize and represent the underlying data in a compressed way. Such a hierarchy is particularly useful for data visualization.

Alternatively, in some applications we may believe that the data bear an underlying hierarchical structure that we want to discover. For example, hierarchical clustering may uncover a hierarchy for *AllElectronics* employees structured on, say, salary. In the study of evolution, hierarchical clustering may group animals according to their biological features to uncover evolutionary paths, which are a hierarchy of species. As another example, grouping configurations of a strategic game (e.g., chess or checkers) in a hierarchical way may help to develop game strategies that can be used to train players. In this section, you will study hierarchical clustering methods. Section 10.3.1 begins with a discussion of agglomerative versus divisive hierarchical clustering, which organize objects into a hierarchy using a bottom-up or top-down strategy, respectively. Agglomerative methods start with individual objects as clusters, which are iteratively merged to form larger clusters. Conversely, divisive methods initially let all the given objects form one cluster, which they iteratively split into smaller clusters. Hierarchical clustering methods can encounter difficulties regarding the selection of merge or split points. Such a decision is critical, because once a group of objects is merged or split, the process at the next step will operate on the newly generated clusters. It will neither undo what was done previously, nor perform object swapping between clusters. Thus, merge or split decisions, if not well chosen, may lead to low-quality clusters. Moreover, the methods do not scale well because each decision of merge or split needs to examine and evaluate many objects or clusters.

3. Apriori Algorithm

3.1Introduction to the Apriori Algorithm

We break the Apriori algorithm into three sections, as illustrated in Figure 1.Initial frequent item sets are fed into the system, and candidate generation, candidatepruning, and candidate support is executed in turn. The support information is fedback into the candidate generator and the cycle continues until the final candidate set is determined. We will first introduce some of the datamining lexicon and thendescribe the operational phases in more detail. In the literature, an analogy to a shopping cart is used: the set of items purchasedat one time, checked out from the library, or otherwise grouped together based onsome criteria such as time, customer, etc. is referred to as a basket. The items withinthe basket can be the entire transaction, or there may be multiple transactions withinthe basket. A frequent itemset is the a set of one or more items that often occur in adatabase one item, and often occurs together in the same basket within the databaseif it consists of more than one item. The cut-off of how often a set must occur beforeit is included in the candidate set is the support.

In this way, a researcher can request a particular support value and find the itemswhich occur together in a basket a minimum number of times within the database, guaranteeing a minimum confidence in the results. A popular example in the literature(possibly apocryphal) is processing the supermarket transactions of working men withyoung children: when they go to the store after work to pick up diapers, they tendto purchase beer at the same time. Thus, it makes sense statistically, if not sociallyresponsibly, to put a beer refrigerator in the diaper aisle.

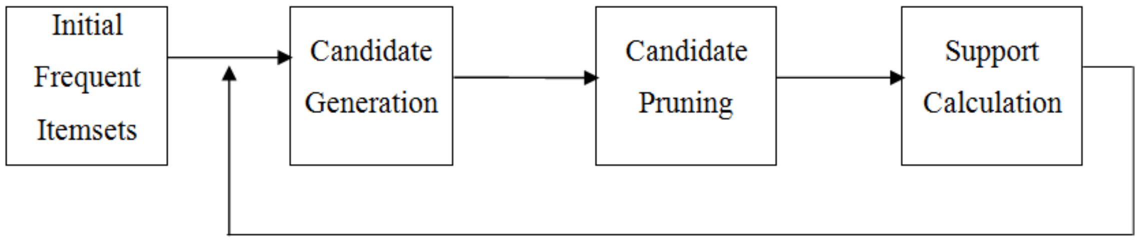


Figure 3.1 Process flow of the data mining system

Candidate generation is the process in which one generation of candidates arebuilt into the next generation. This building process is from where the Apriori namederives. Because each new candidate is built from candidates that have been determined apriori (in the previous generation) to have a high level of support, theycan be confidently expanded into new potential frequent itemsets. This is expressedformally as follows:

```
$\forall f_1, f_2 \varepsilon F_k$do
with$f_1 = (i_1, .... , i_{k-1}, i_k)$
and$f_2 = (i_1, .... , i_{k-1}, i^*_k)$
and$i_k < i^*_k$
$f := f1 \ U \ f2 = (i_1, .... , i_{k-1}, i_k, i^*_k)$
```

It should be noted that only ordered sets are utilized. Thus, when f is generatedfrom f1 and f2, the sets remain ordered. Candidate generation pairs up any candidates that differ only in their final element to generate the next candidate generation.The next step of candidate generation guarantees that each new candidate is notonly formed from two candidates from the previous generation, but that every subsetof it is also present in the previous generation, as follows:

$\forall i \ \varepsilon f : f - \{i\} \varepsilon \ F_k$

Thus, our initial candidate generation proves by design that if we remove eitherof the last two items (i_k, i^*_k) from the new candidate, we will get candidates from theprevious generation, namely, f_1 and f_2. The second step proves that if we removeany of the other items from the new candidate, we must find a candidate from theprevious generation. This progressive build-up of candidates is the heart of the Apriorialgorithm.The third phase of the algorithm is the support calculation. It is by far themost time consuming and data intensive part of the application, as it is during thisphase the database is streamed into the system. Each potential candidate's support,or number of occurrences over the database set, is determined by comparing eachcandidate with each transaction in the database. If the set of items that make upthe candidate appear in the transaction, the support count for that candidate isincremented, as follows:

```
$\forall t \ \varepsilon \ T$ do
    $\forall c \ \varepsilon \ C$ do
        if $c \ \varepsilon \ t$
            support$(c)$++
```

The main problem with the Apriori algorithm is this data complexity. Eachcandidate must be compared against every transaction data, and candidate generationmust see the entire database transaction set. This gives a large running time for asingle generation, $O(|T||C||t|)$, assuming the subset function can be implemented inconstant time $|t|$. However, the parallelism contained in the loops allows for someinteresting acceleration in hardware, particularly when implemented as a systolicarray.

3.2 The Apriori Algorithm: Basics

The Apriori Algorithms an influential algorithm for mining frequentitemsets for boolean association rules.

Key Concepts

- Frequent Itemsets: The sets of item which has minimum support (denoted by $L_i$for ith-Itemset).
- Apriori Property: Any subset of frequent itemset must be frequent.
- Join Operation: To find L_k, a set of candidate k-itemsets is generated by joining $L_{k-1}$with itself.

3.3 The Apriori Algorithm in a Nutshell

- Find the *frequent itemsets*: the sets of items that have minimum support
- A subset of a frequent itemset must also be a frequent itemset
- i.e., if $\{AB\}$ isa frequent itemset, both $\{A\}$ and $\{B\}$ should be a frequent itemset
- Iteratively find frequent itemsets with cardinality from 1 to *k* (*k*-itemset)
- Use the frequent itemsets to generate association rules.

3.4 Minimum Support Threshold

The support of an association pattern is the percentage of task-relevant data transactions for which the pattern is true.
IF A $\Rightarrow$ B

$$support\,(A \Rightarrow B) = \frac{no.\,of\ tuples\ containing\ both\ A\ and\ B}{total\ no.\,of\ tuples}$$

3.5 Minimum Confidence Threshold

Confidence is defined as the measure of certainty ortrustworthiness associated with each discovered pattern.
IF A ⇒ B

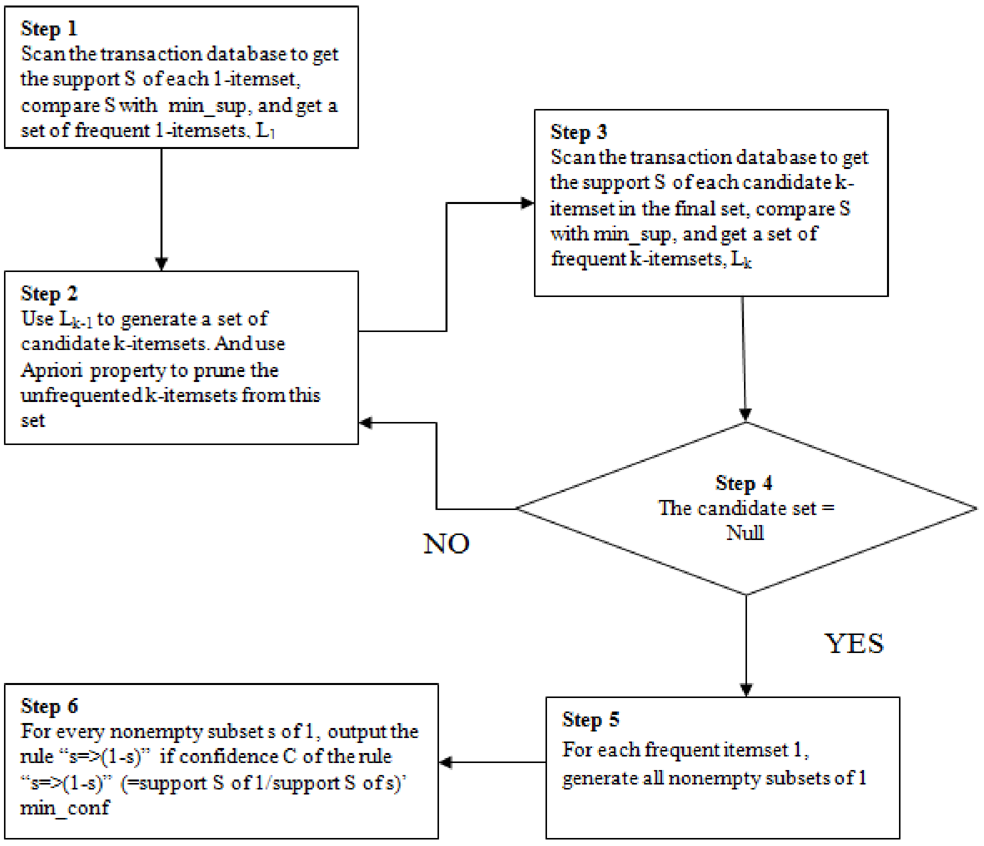
$$confidence(A \Rightarrow B) = \frac{no.\,of\ tuples\ containing\ both\ A\ and\ B}{no.\,of\ tuples\ containing\ A}$$

3.6 The Apriori Algorithm: Pseudo code

- Join Step: $C_k$is generated by joining $L_{k-1}$with itself
- Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset

- Pseudo-code:

```
    $C_k$: Candidate itemset of size k
    $L_k$: frequent itemset of size k
    $L_1$= {frequent items};
    for($k$= 1; $L_k$!=  ; $k$++) do begin
        $C_{k+1}$= candidates generated from $L_k$;
    for eachtransaction $t$in database do
        increment the count of all candidates in $C_{k+1}$that are contained in $t$
        $L_{k+1}$= candidates in $C_{k+1}$with min_support
end
return$\cup_k L_k$;
```


Step 1
Scan the transaction database to get the support S of each 1-itemset, compare S with min_sup, and get a set of frequent 1-itemsets, L1
Step 2
Use Lk-1 to generate a set of candidate k-itemsets. And use Apriori property to prune the unfrequented k-itemsets from this set
Step 3
Scan the transaction database to get the support S of each candidate k-itemset in the final set, compare S with min_sup, and get a set of frequent k-itemsets, Lk
Step 4
The candidate set = Null
NO
YES
Step 5
For each frequent itemset 1, generate all nonempty subsets of 1
Step 6
For every nonempty subset s of 1, output the rule "s=>(1-s)" if confidence C of the rule "s=>(1-s)" (=support S of 1/support S of s)' min_conf

3.7 The Apriori Algorithm: Example

TID	List of Items
T100	I1, I2, I5
T100	I2, I4
T100	I2, I3
T100	I1, I2, I4
T100	I1, I3
T100	I2, I3
T100	I1, I3
T100	I1, I2 ,I3, I5
T100	I1, I2, I3

- Consider a database, D , consisting of 9 transactions.
- Suppose min. support count required is 2 (i.e. min_sup = 2/9 = 22 %)
- Let minimum confidence required is 70%.
- We have to first find out the frequent itemset using Apriori algorithm.
- Then, Association rules will be generated using min. support & min. confidence.

Step 1. Generating 1-itemset Frequent Pattern

Scan D for count of each candidate

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Compare candidate support count with minimum support count

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

C_1 **L_1**

- The set of frequent 1-itemsets, L1, consists of the candidate 1-itemsets satisfying minimum support.
- In the first iteration of the algorithm, each item is a member of the set of candidate.

Step 2. Generating 2-itemset Frequent Pattern

Generate C_2 candidate from L_1

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

Scan D for count of each candidate

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

Compare candidate support count with minimum support count

Itemset	Sup Count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

- To discover the set of frequent 2-itemsets, L2, the algorithm uses L1 *Join* L1to generate a candidate set of 2-itemsets, C_2.
- Next, the transactions in D are scanned and the support count for each candidate itemset in C2is accumulated (as shown in the middle table).
- The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate 2-itemsets in C2having minimum support.
- Note: We haven't used Apriori Property yet.

Step 3: Generating 3-itemset Frequent Pattern

- The generation of the set of candidate 3-itemsets, C3, involves use of the Apriori Property.
- In order to find C3, we compute L2*Join*L2.
- C3= L2 *Join*L2 = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}.

- Now, Join step is complete and Prune step will be used to reduce the size of C3. Prune step helps to avoid heavy computation due to large Ck.
- Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that four latter candidates cannot possibly be frequent. How ?
- For example, let's take {I1, I2, I3}.The 2-item subsets of it are {I1, I2}, {I1, I3} & {I2, I3}. Since all 2-item subsets of {I1, I2, I3} are members of L2, We will keep {I1, I2, I3} in C3.
- Lets take another example of {I2, I3, I5}which shows how the pruning is performed. The 2-item subsets are {I2, I3}, {I2, I5} & {I3,I5}.
- BUT, {I3, I5} is not a member of L2and hence it is not frequent violating Apriori Property. Thus we will have to remove {I2, I3, I5} from C3.
- Therefore, C3= {{I1, I2, I3}, {I1, I2, I5}} after checking for all members of result of Join operation for Pruning.
- Now, the transactions in D are scanned in order to determine L3, consisting of those candidates 3-itemsets in C3having minimum support.

Step 4: Generating 4-itemset Frequent Pattern

- The algorithm uses L3 *Join*L3to generate a candidate set of 4-itemsets, C4. Although the join results in {{I1, I2, I3, I5}}, this itemset is pruned since its subset {{I2, I3, I5}}is not frequent.
- Thus, C4= φ, and algorithm terminates, having found all of the frequent items. This completes our Apriori Algorithm.
- What's Next?

These frequent itemsets will be used to generate strong association rules(where strong association rules satisfy both minimum support & minimum confidence).

Step 5: Generating Association Rules from Frequent Itemsets

- Let minimum confidence threshold is , say 70%.
- The resulting association rules are shown below, each listed with its confidence.

–R1: I1 ^ I2 →I5
- Confidence = sc{I1,I2,I5}/sc{I1,I2} = 2/4 = 50%
- R1 is Rejected.

–R2: I1 ^ I5 →I2
- •Confidence = sc{I1,I2,I5}/sc{I1,I5} = 2/2 = 100%
- •R2 is Selected.

–R3: I2 ^ I5 →I1
- •Confidence = sc{I1,I2,I5}/sc{I2,I5} = 2/2 = 100%
- •R3 is Selected.

–R4: I1 →I2 ^ I5
- •Confidence = sc{I1,I2,I5}/sc{I1} = 2/6 = 33%
- •R4 is Rejected.

–R5: I2 →I1 ^ I5
- •Confidence = sc{I1,I2,I5}/{I2} = 2/7 = 29%
- •R5 is Rejected.

–R6: I5 →I1 ^ I2
- •Confidence = sc{I1,I2,I5}/ {I5} = 2/2 = 100%
- •R6 is Selected.

In this way, we have found three strong association rules.

3.8 Methods to Improve Apriori's Efficiency

- Hash-based itemset counting: A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent.
- Transaction reduction: A transaction that does not contain any frequent k-itemsetis useless in subsequent scans.
- Partitioning: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB.
- Sampling: mining on a subset of given data, lower support threshold + a method to determine the completeness.
- Dynamic itemset counting: add new candidate itemsets only when all of their subsets are estimated to be frequent.

4. Association Rule

4.1 Data Mining

Data Mining refers to extracting or mining information fromlarge amounts of data. Data mining has attracted a great deal ofattention in the information industry and in society as a whole inrecent years, due to the wide availability of huge amounts ofdata and the imminent need for turning such data into usefulinformation and knowledge. Data mining, "*the extraction of hidden predictive informationfrom large databases"*, is a powerful new technology with greatpotential to help companies focus on the most importantinformation in their data warehouses. Data mining tools predictfuture trends and behaviours, allowing businesses to makeproactive, knowledge-driven decisions. The automated, prospective analysis offered by data miningmove beyond the analysis of past events provided byretrospective tools typical of decision support systems. Datamining tools can answer business questions that traditionallywere too time consuming to resolve. They scour databases forhidden patterns, finding predictive information that experts maymiss because it lies outside their expectations. Most companies collect and refine massive quantities of data.Data mining techniques can be implemented rapidly on existingsoftware and hardware platforms to enhance the value ofexisting information resources and can be integrated with newproducts and systems as they are brought on-line. Whenimplemented on high performance client/server or parallelprocessing computers, data mining tools can analyze massivedatabases to deliver answers to many questions.

The information and knowledge gained can be used forapplication ranging from market analysis, fraud detection, andcustomer retention, to production control and scienceexploration. Data Mining plays an important role in onlineshopping for analyzing the subscribers" data and understandingtheir behaviours and making good decisions such that customeracquisition and customer retention are increased which giveshigh revenue.

4.2 Association Rule Mining

Association Rule Mining is a popular and well researchedmethod for discovering interesting relations between variables inlarge databases. Association rules are statements of the form{X1, X2, …, Xn} => Y meaning that if all of X1, X2,… Xn isfound in the market basket, and then we have good chance offinding Y. the probability of finding Y for us to accept this ruleis called the confidence of the rule. Normally rules that have aconfidence above a certain threshold only will be searched. Inmany situations, association rules involves sets of items thatappear frequently. For example, a good marketing strategycannot be run involving items that no one buys. Thus, much datamining starts with the assumption that sets of items with supportare only considered.The discovery of such associations can help retailers developmarketing strategies by gaining insight into which items arefrequently purchased together by customer and which itemsbring them better profits when placed with in close proximity.The two types of finding association between products existingin a large database are Boolean and Quantitative. Booleanassociation rule mining finds association for the entire dataset.Quantitative association rule mining finds association for theclusters formed from the dataset.

4.3 Prediction

Data mining automates the process of finding predictive information in large databases. Questions that traditionally required extensive hands-on analysis can now be answered directly from the data quickly. The primary task of association mining is to detect frequently co-occurring groups of items in transactional databases. The intention is to use this knowledge for prediction purposes.

Early attempts for prediction used classification and performance was favourable. In this project, any item is allowed to be treated as a class label its value is to be predicted based on the presence of other items. Put another way, knowing a subset of the shopping carts contents, we want to "guess" (predict) the rest. Suppose the shopping cart of a customer at the checkout counter contains bread, butter, milk, cheese, and pudding. Could someone who met the same customer when the cart contained only bread, butter, and milk, have predicted that the person would add cheese and pudding? It is important to understand that allowing any item to be treated as a class label presents serious challenges as compared with the case of just a single class label. The number of different items can be very high, perhaps hundreds, or thousand, or even more. To generate association rules for each of them separately would give rise to great many rules with two obvious consequences: first, the memory space occupied by these rules can be many times larger than the original database (because of the task's combinatorial nature); second, identifying the most relevant rules and combining their sometimes conflicting predictions may easily incur prohibitive computational costs. In this work, both of these problems are solved by developing a technique that answers user's queries (for shopping cart completion) in a way that is acceptable not only in terms of accuracy, but also in terms of time and space complexity.

This paradigm can be exploited in diverse applications. For example, in the each "shopping cart" contained a set of hyperlinks pointing to a Web page; in medical applications, the shopping cart may contain a patient's symptoms, results of lab tests, and diagnoses; in a financial domain, the cart may contain companies held in the same portfolio.In all these databases, prediction of unknown items can play a very important role. For instance, a patient's symptoms are rarely due to a single cause; two or more diseases usually conspire to make the person sick. Having identified one, the physician tends to focus on how to treat this single disorder, ignoring others that can meanwhile deteriorate the patient's condition. Such unintentional neglect can be prevented by subjecting the patient to all possible lab tests. However, the number of tests one can undergo is limited by such practical factors as time, costs, and the patient's discomfort. A decision support system advising a medical doctor about which other diseases may accompany the ones already diagnosed can help in the selection of the most relevant additional tests.

4.4 Dempster's Rule of Combination

Dempster's rule of combination (DRC) is used to combine the discovered. When searching for a way to predict the presence of an item in partially observed shopping carts, association rules are used. However, many rules with equal antecedents differ in their consequents and some of these consequents contain the desired item to be predicted, others do not. The question is how to combine (and how to quantify) the potentially conflicting evidences. DRC is used for this purpose. Finally the predicted items are suggested to the user.

5. Literature Survey

5.1. Mining association rules using new support and confidence:

Item set evaluation, in classical association rule mining, by support was based on counting. A link based measure termed as w-support is used to formulate association rule mining. This system is called as "w-support link based "because w-support can be regarded as a generalization of support. It takes weights, are not determined by assigning values to items, of transactions for evaluation. Therefore this approach is more effective than counting based measurement.

5.2. Alarms Association Rules

Hou Sizu introduced an association analysis model which consists of seven components namely:

- Alarms databases – A repository of underlying effective alarms data.
- Data Import – Alarm Tables are imported from alarms database to mining database for mining alarms.
- Data Pre-processing – Conversion of data tables into the mining unified data format takes place.
- Alarms Correlation – Alarms correlation analysis such as alarms compression, alarms filtering, alarms count, etc are used to convert and compress alarms.
- Sequential Pattern Mining – A sequential pattern mining algorithm is used to mine the selected unified format data in the mining database.
- Post Processing – The main function of this module is to compile the results of mining into a single form like grouping, sorting and conversion for rules.
- Expert Evaluation and Data Testing – Refining the mining results by adjusting the parameters for next iteration.
- Rules Knowledge Warehouse – It serves as a storage of alarms correlation rules which are mined by rules engines.

This model can be used to assist network managers to position the fault quickly and accurately.

5.3. Realization of association matrix mining algorithm:

Wang describes a transaction database as $D=B^{(0)}$. I , where D models transaction set, I models item set, Matrix $B^{(0)}$is the transaction item association matrix. The elements can be defined as: to transaction I, if it associates with item j, then the corresponding element will be 1 otherwise 0.

$$D_i = \sum_{j=1}^{N} Bij^{(0)} I_i \qquad i=1,2,....,N$$

Wherei=

1,2,…,N is a transaction set.
J=1,2,...,M is a item set.

The frequency of the j^th^item appears in the whole transaction matrix is determined by Lj,

$$L_j = \sum_{i=1}^{N} Bij^{(0)}$$

Elementary operations are done to the matrix by adding the $B^{(0)}$ of the item sets to get the new matrix $B^{(1)}$so that it is easy to identify frequent pairs of items. Later based on minimum support value, it is extended to more combination of items. At last, we could get the frequent occurring group of items satisfying minimum support. This paper realizes the formation of candidate item sets of Apriori algorithm by elementary matrix operations. It increases the efficiency of data mining.

5.4. Apriori algorithm for Tax inspection excavation:

Qing – Xiang Zhu applies the apriori algorithm of association rules of data mining into tax inspection cases to accurately identify the dishonest enterprise in order to improve the efficiency and effectives of inspection. The prediction rules worked with some input information about the enterprises to which tax evasion occurred to the system.

5.5. Demand of Data mining for traffic Management

Wei Cheng describes data pre-processing based on Anomaly Detection via analysis of traffic violate type, analysis of the composition of the traffic accidents and weather , time, road type and analyse the situation of the traffic violation and drivers. Association rules was implemented in the above analysis for effective traffic management. Apriori algorithm used the database report generated from the above analysis as input for it's working.

5.6. Association Rule Mining using Coherent rules

Tejaswi Pinniboyina proposed a new coherent rules algorithm for association rules. This algorithm which is based on coherent rules allows users to mine the data without the domain knowledge. The results are comparatively outstanding over the performance of basic association rules without coherent principle.

5.7. A Boolean Matrix algorithm for association rule mining

Y Jaya Babu has introduced a new boolean matrix algorithm for effective spatial data mining based on association rules. The efficiency of extracting spatial association rules was outstanding when compared to normal Apriori algorithm. This algorithm has reduced the number of times of scanning the transaction database and decreased the number of the set of candidate itemsets.

6. SYSTEM DESIGN

Any project developed today is said to be good only if it hassome basic characteristics such as modularity, loose couplingand high cohesion. A component is classified as good qualityonly if it is modular, loosely coupled and has high cohesion i.e.,each component should be independent of the other and eachcomponent must be focused only on its particular purpose.Finally the component should be modular so that thedevelopment of the components is understandable, can easily beenhanced in the future and also easy to locate and correct errorswithout affecting the other components involved in the project.

The following sections deals with how this system is designed,the modules involved and overall architecture diagram of thesystem which shows the modules present in the system.

6.1 Shopping Cart Prediction Architecture

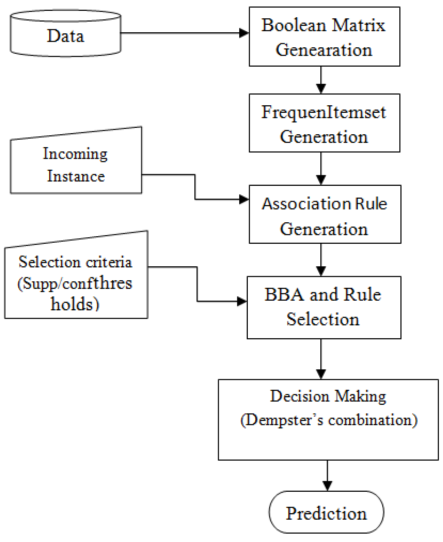

Data
Boolean Matrix Geneartion
FrequenItemset Generation
Incoming Instance
Association Rule Generation
Selection criteria (Supp/confthres holds)
BBA and Rule Selection
Decision Making (Dempster's combination)
Prediction

Fig.6.1. Shopping cart prediction architecture

Fig.6.1. shows the shopping cart prediction architecture in whichthe Boolean Matrix is generated by transforming the databaseinto Boolean values. The frequent itemsets are generated fromthe Boolean matrix. At this stage we need the Support value.Then association rules are to generated from the alreadygenerated frequent itemsets. It takes minimum confidence fromthe user and discovers all rules with a fixed antecedent and withdifferent consequent. The association rules generated form thebasis for prediction.

We assign BBA value to each association rule generated.This gives more weight to rules with higher support masses areassigned based on both their confidence and support values. The incoming itemset i.e. the content of incoming shopping cartwill also be represented by a Boolean vector and AND operationis performed with each transaction vector to generate theassociation rules. Finally the rules are combined to get thepredictions. Dempster's rule of combination (DRC) is used to combinethe evidences. When searching for a way to predict the presenceor absence of an item in a partially observed shopping cart s, wewanted to use association rules. However, many rules with equal antecedents differ in theirconsequents—some of these consequents contain the desireditem to be predicted, others do not. The question is how tocombine (and how to quantify) the potentially conflicting evidences. DRC is used for this purpose. Finally thepredicted items are suggested to the user.

7. IMPLEMENTATION

This topic consists of detailed description of each and everymodule with its advantages and data and execution flow of eachmodule with algorithm. It helps to understand each and everymodule of the project more deeply and clearly. Each descriptionconsists of the basic concept of the module, input and also theexcepted output.

7.1 Modules

The project has been divided into various modules and eachmodule has been completed within a scheduled time line. Thefollowing are the modules of the project are boolean matrixgeneration,Frequent itemset generation, Association rulegeneration, BBA and decision making.

7.2 Boolean Matrix Generation

This module is to convert the data's in the database and theincoming instance to database into Boolean value (either 0's or1's). If an item is present in the transaction it is marked with theBoolean value 1 else the item is marked as 0. Raw database"**rdb**" is a m x n matrix where 'm' is the number oftransactions and 'n' is the number of attributes. By using abovementioned rule the Raw database in converted into Booleandatabase "**bdb**"(rdb [i, j] => bdb [i, j] where 'i' represent therows and 'j' represent the columns).

```
The algorithm used is given here.
for all i<=m do
for all j<=n do
if jth item is present in ith row
set rdb[i,j] =1
else
set rdb[i,j] =0
end do
end do
```

The example input database given to this stage is shownhere. The database contains eight transactions and seven items.In real world this will be of large size but this is used for illustration purpose.

Table.7.1. Example Input Database

Table							
ID	Field1	Field2	Field3	Field4	Field5	Field6	Field7
7	N	N	N	N	N	N	Y
8	Y	Y	N	N	Y	Y	N
9	Y	Y	N	N	Y	Y	Y
10	N	N	Y	Y	Y	Y	N
11	N	N	N	Y	Y	Y	N
12	Y	Y	N	N	N	Y	Y
13	N	N	Y	Y	Y	Y	Y
14	N	N	Y	Y	Y	Y	Y
15	N	N	Y	Y	Y	Y	N

Table.7.1 shows that If an item is contained in a transaction, thecorresponding attribute value will be 'Y', otherwise the valuewill be 'N'.The output of this module will be the Booleanmatrix, which will look as this.

$$\begin{array}{ccccccc}
1 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 0
\end{array}$$

If an item is contained in a transaction, the correspondingattribute value will be 1; otherwise the value will be 0.

7.3 Frequent Itemset Generation

This module finds out the frequent item set from the existingtransaction based on the support value. It involves Join step andPrune step. This module takes the input from the previous stageand forms the frequent itemset from matrix table whose valuesare 1 for transaction. This module also generates the Booleanvector for the frequent item set along with support value.Boolean vector takes the value 'true' for the item present in the itemset and takes the value 'null' for the item not present in theitemset.
The algorithm used is shown here. It has two steps as explainedabove.
for each column ci of pdb

```
if sum(ci) >= new_support
f1 = ii
else delete ci from pdb
for each row rj of pdb
if sum(rj) < 2
delete rj from pdb
for (k=2;| fk-1|>k-1;k++)
{
produce k-vectors combination for all columns of
bdb;
for each k-vectors combination {$c_{i1}$,$c_{i2}$, $c_{i3}$ … ,$c_{ik}$ }
{
b= ci1 • ci2 •….•cik
if sum(b)>= new_support
fk={ ii1, ii2,……,iik };
}
for each item ii in fk
if | fk(ii)| < k
delete the column ci according to item ii from bdb;
for each row rj from bdb
if sum(rj) < k+1
delete rj from bdb;
k=k+1
}
return f= f1 u f2 … u fk
```

The input given to this stage is the Boolean matrixgenerated in previous module. This is the sample input that isgiven for illustration. The support value is also given as input.

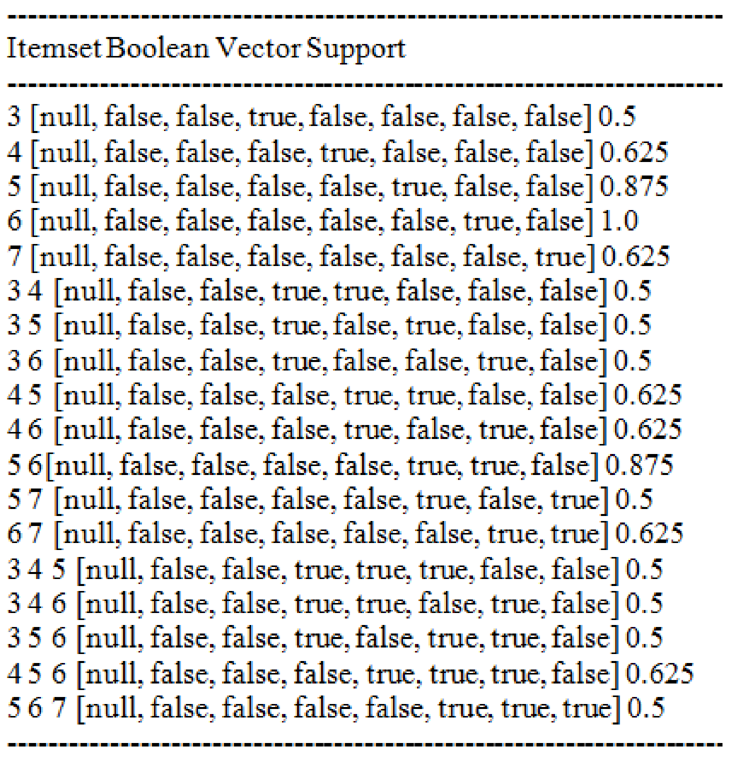
1	1	0	0	1	1	0
1	1	0	0	1	1	1
0	0	1	1	1	1	0
0	0	0	1	1	1	0
1	1	0	0	0	1	1
0	0	1	1	1	1	1
0	0	1	1	1	1	1
0	0	1	1	1	1	0

Support Value: 50

The output of this stage is the frequent itemsets generated fromBoolean matrix. Each frequent itemset generated is alsoconverted into a Boolean vector as shown in the table below.

```
Input configuration: 7 items, 8 transactions, minsup =
50.0%
Frequent 1-itemsets
[3, 4, 5, 6, 7]
Frequent 2-itemsets
[3 4, 3 5, 3 6, 4 5, 4 6, 5 6, 5 7, 6 7]
Frequent 3-itemsets
[3 4 5, 3 4 6, 3 5 6, 4 5 6, 5 6 7]
Frequent 4-itemsets
[3 4 5 6]
```

Table.7.2. Frequent itemsets generation output



```
---------------------------------------------------------------------
ItemsetBoolean Vector Support
---------------------------------------------------------------------
3 [null, false, false, true, false, false, false, false] 0.5
4 [null, false, false, false, true, false, false, false] 0.625
5 [null, false, false, false, false, true, false, false] 0.875
6 [null, false, false, false, false, false, true, false] 1.0
7 [null, false, false, false, false, false, false, true] 0.625
3 4  [null, false, false, true, true, false, false, false] 0.5
3 5  [null, false, false, true, false, true, false, false] 0.5
3 6  [null, false, false, true, false, false, true, false] 0.5
4 5  [null, false, false, false, true, true, false, false] 0.625
4 6  [null, false, false, false, true, false, true, false] 0.625
5 6[null, false, false, false, false, true, true, false] 0.875
5 7  [null, false, false, false, false, true, false, true] 0.5
6 7  [null, false, false, false, false, false, true, true] 0.625
3 4 5 [null, false, false, true, true, true, false, false] 0.5
3 4 6 [null, false, false, true, true, false, true, false] 0.5
3 5 6 [null, false, false, true, false, true, true, false] 0.5
4 5 6 [null, false, false, false, true, true, true, false] 0.625
5 6 7 [null, false, false, false, false, true, true, true] 0.5
---------------------------------------------------------------------
```

Table.6.2. shows the frequent itemsets along with the BooleanVector generated from Boolean Matrix. User Support value isneeded for this. If an item is present in the frequent itemset, ittakes the value 'True' in the Boolean Vector, otherwise thevalue will be 'False'. It involves Join step and Prune step. Thefrequent itemset generated should have desired support value.

7.4 Association Rule Generation

This module is used to generate association rules from thealready generated frequent itemsets.The algorithm uses the factthat:

"If there exists two rules A->C and A->{C U X} where Xdoesn't belongs to A U C then the confidence of the secondcannot be larger than the first one". The algorithm checks if a given set is a subset of another set ornot. To perform this operation each item in an itemset is represented as an integer where a bit corresponding to as item isset to 1. For example, suppose a database with 8 attributes, itemset {1,2,5} is represented as 38 as follows. 0 0 1 0 0 1 1 0

To check if set {2,5} is a subset of {1,2,5} we represent {2,5}like above and is evaluated to 36. Now we perform ANDoperation 38 & 36. The result is checked for equality with thefirst itemset ({2, 5}). If they are equal then it is a subsetotherwise it is not. In this case the result is obvious. Similarlydifference of two sets is done during production of the rules. This algorithm is capable of finding all association rules with afixed antecedent and with different consequents from thefrequent itemsets subject to a user specified minimumconfidence very quickly.It takes minimum confidence from the user and discovers allrules with a fixed antecedent and with different consequent. Thismodule also takes the frequent item set and the incomingshopping cart instance to generate the association rule with thecorresponding support and confidence value.

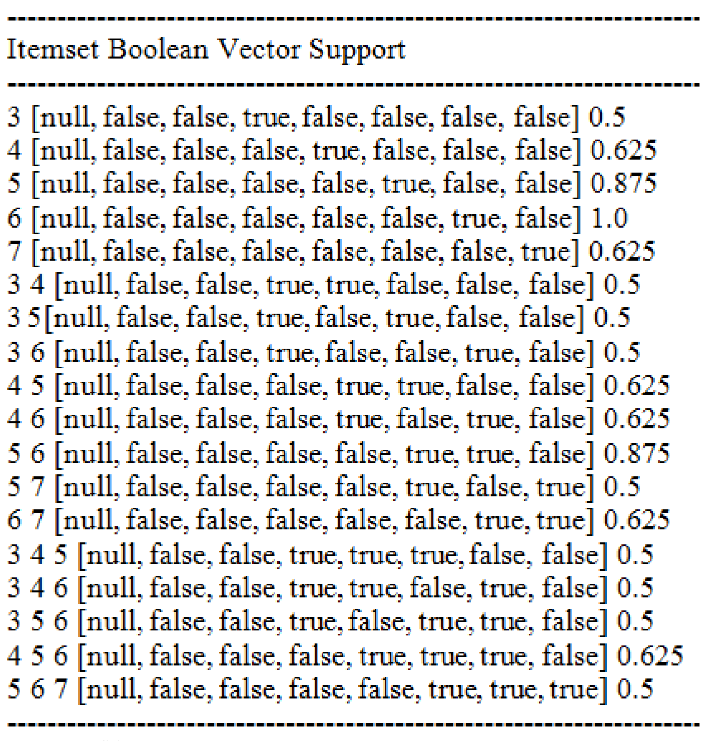
The algorithm used is shown here.

```
for all fk, fk ∈F, 1<=k<=maxsize-1 do begin
rsup=support(fk)*miconf
found=0
for all fm, fm _ Fk +1<= m <=maxsize do begin
if (support(fm)>=rsup) then begin
if(fk ⊂ fm) then begin
found=found+1
conf=support(fm)/ support(fk)
generate the rule fk = (fm - fk) &= conf and
support=support(fm)
end if
else
if (found<2)
```

```
continue step1 with next k
else found=0
endif
endif
end do
end do
```

The input given to the rule generation process is the Booleanvectors representing each transaction and also the contents ofincomplete shopping cart. This is shown in Table 6.3.

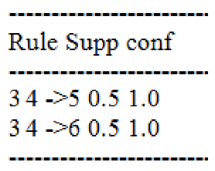
Table.7.3. Rule Generation Input



```
----------------------------------------------------------------------
Itemset Boolean Vector Support
----------------------------------------------------------------------
3 [null, false, false, true, false, false, false, false] 0.5
4 [null, false, false, false, true, false, false, false] 0.625
5 [null, false, false, false, false, true, false, false] 0.875
6 [null, false, false, false, false, false, true, false] 1.0
7 [null, false, false, false, false, false, false, true] 0.625
3 4 [null, false, false, true, true, false, false, false] 0.5
3 5[null, false, false, true, false, true, false, false] 0.5
3 6 [null, false, false, true, false, false, true, false] 0.5
4 5 [null, false, false, false, true, true, false, false] 0.625
4 6 [null, false, false, false, true, false, true, false] 0.625
5 6 [null, false, false, false, false, true, true, false] 0.875
5 7 [null, false, false, false, false, true, false, true] 0.5
6 7 [null, false, false, false, false, false, true, true] 0.625
3 4 5 [null, false, false, true, true, true, false, false] 0.5
3 4 6 [null, false, false, true, true, false, true, false] 0.5
3 5 6 [null, false, false, true, false, true, true, false] 0.5
4 5 6 [null, false, false, false, true, true, true, false] 0.625
5 6 7 [null, false, false, false, false, true, true, true] 0.5
----------------------------------------------------------------------
```

```
Enter confidence: 80
Enter incoming shopping cart contents: 3 4
```

Table.7.4. Rule Generation Output



```
------------------------
Rule Supp conf
------------------------
3 4 ->5 0.5 1.0
3 4 ->6 0.5 1.0
------------------------
```

Table.6.4. shows the output rules generated along with thesupport and confidence values.

7.5 BBA And Decision Making

7.5.1 PARTITIONED-SUPPORT

In many applications, the training data set is skewed. Thus, in asupermarket scenario, the percentage of shopping cartscontaining, say canned fish, can be 5 percent, the remaining 95percent shopping carts not containing this item. Hence, the rulesthat suggest the presence of canned fish will have very lowsupport while rules suggesting the absence of canned fish willhave a higher support.

Unless compensated for, a predictor built from a skewed trainingset typically tends to favour the “majority” classes at the expenseof “minority” classes. In many scenarios, such a situation mustbe avoided.To account for this data set skewness, we propose to adopt a modified support value termed partitioned-support.

The partitioned-support p_supp of the rule, r (a) -> r(c), is thepercentage of transactions that contain r (a) among thosetransactions that contain r(c), i.e.

p_supp = support($r^{(a)}$ U $r^{(c)}$) / support($r^{(c)}$)

7.5.2 BBA:

In association mining techniques, a user-set minimum supportdecides about which rules have “high support.” Once the rulesare selected, they are all treated the same, irrespective of how high or how low their support. Decisions are then made solelybased on the confidence value of the rule. However, a moreintuitive approach would give more weight to rules with highersupport. Therefore, we use a novel method to assign to the rulesmasses based on both their confidence and support values. Thisweight value is called Basic Belief Assignment (BBA).We assign BBA value to each association rule generated.

β = ((1+α2) x conf x p_supp) / (α2 x conf + p_supp); α € [0,1];

Dempster’s rule of combination (DRC) is used to combinethe evidences. When searching for a way to predict the presenceor absence of an item in a partially observed shopping carts, wewanted to use association rules. However, many rules with equalantecedents differ in their consequents—some of theseconsequents contain the desired item to be predicted, others donot. The question is how to combine (and how to quantify) thepotentially conflicting evidences. DRC is used for thispurpose. Some illustrations used from DRC are explained infollowing paragraph.

We remove the overlapping rules while keeping the highestconfidence rule. If two overlapping rules have the sameconfidence, the rule with the lower support is dropped. Finallythe best rule is selected by comparing the mass values. Thepredicted item is then suggested to the user.The input given to this stage is the set of rules generated alongwith their support and confidence values as shown in Table 6.5.

Table.7.5. BBA and Decision Making Input

```
----------------------
Rule Supp conf
----------------------
3 4 ->5 0.5 1.0
3 4 ->6  0.5 1.0
----------------------
```

7.6 Comparison

The performance of both the existing tree approach and theproposed approach is analyzed with databases of different sizes.The results found are very much surprising in the proposedapproach compared to the tree approach. The time of predictionhas decreased to a great extent compared to existing treeapproach.

Table.6.7. Execution Time Comparison

No. of Transaction	Tree Approach	Proposed Approach
	Execution Time	Execution Time
100	48.7	0.797
80	44.172	0.625
60	42.031	0.578
40	40.015	0.593
20	38.721	0.547

Table.7.7. shows the comparison of execution time between theexisting tree approach and proposed approach for differenttransactions.

7.7 Performance Evaluation

The Fig.6.1. shows the performance evaluation graph whichcompares the performance of both the existing tree approach andproposed approach and displays the time taken to execute fordifferent transactions in seconds.

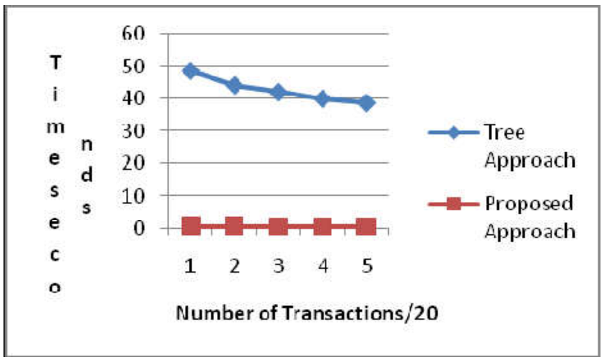


Fig.7.1. Performance Evaluation Graph

Fig 7.2 is the screen shot of the implemented program. Here there are two text box are there.

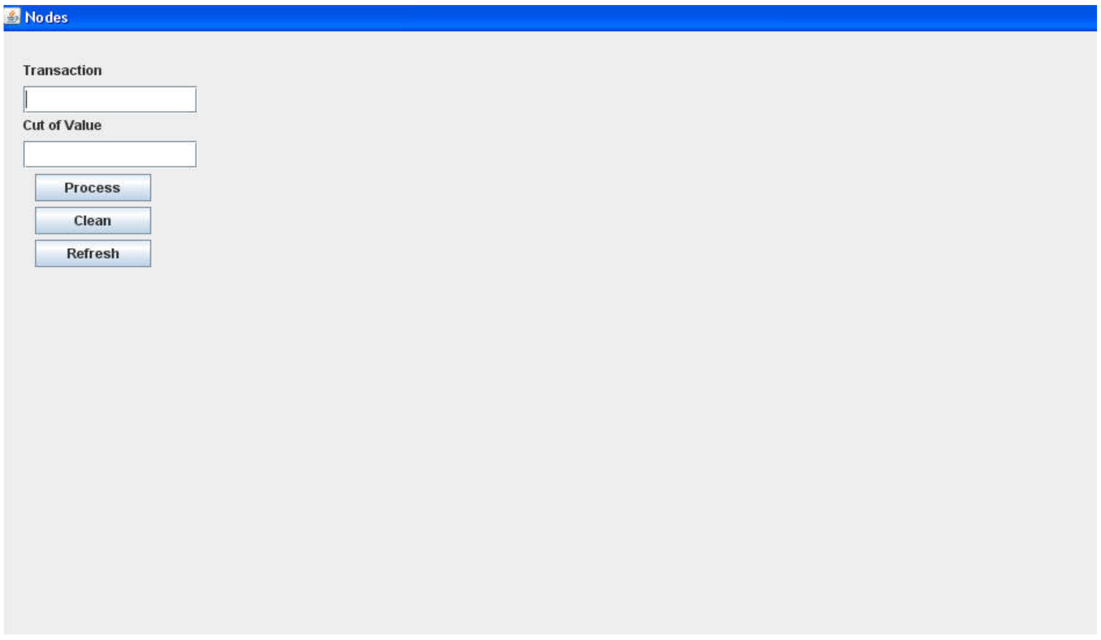


Fig.7.2. Output screenshots

In the Transaction text box we will provide the list of item to be purchased. In the cut of value text box we provide the cut of value that would be identify the transaction pattern which is frequently purchased. Upon which we will predict on the customer or the things to be purchased.

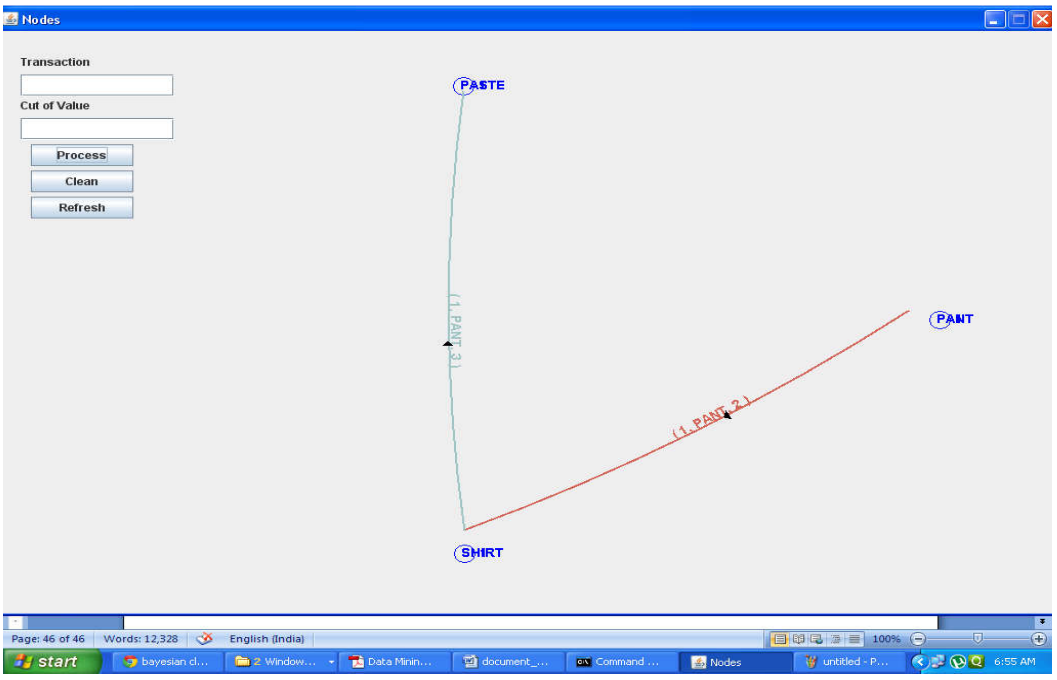


Fig.7.3. Output screenshots

In the Fig 7.3 (pant, shirt, paste) pattern is given. It will show a graph shown in the Fig. Here (1,PANT,2) indicate like following:

1:- no of occurrence of the pattern

PANT: starting node

2: pattern consisting of nodes

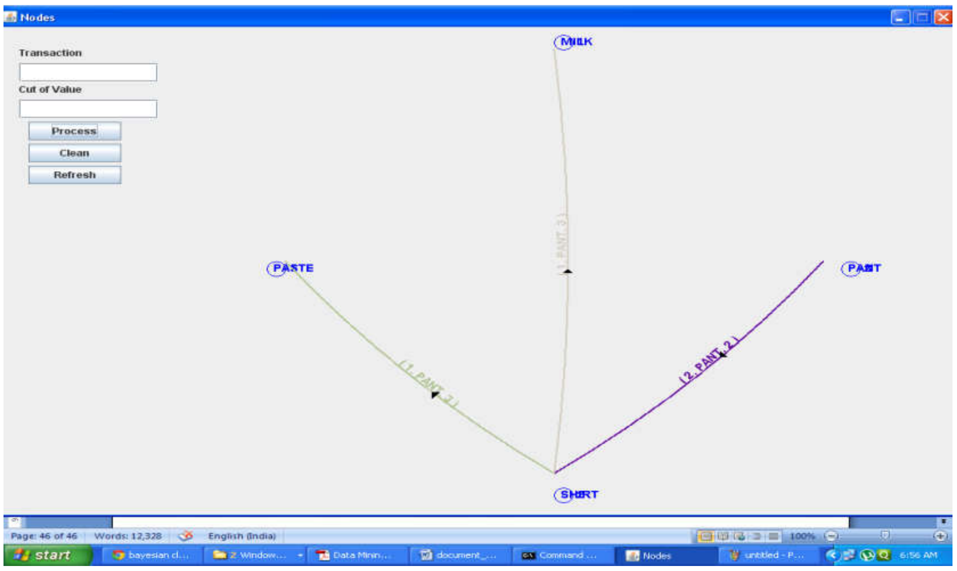


Fig.7.4. Output screenshots

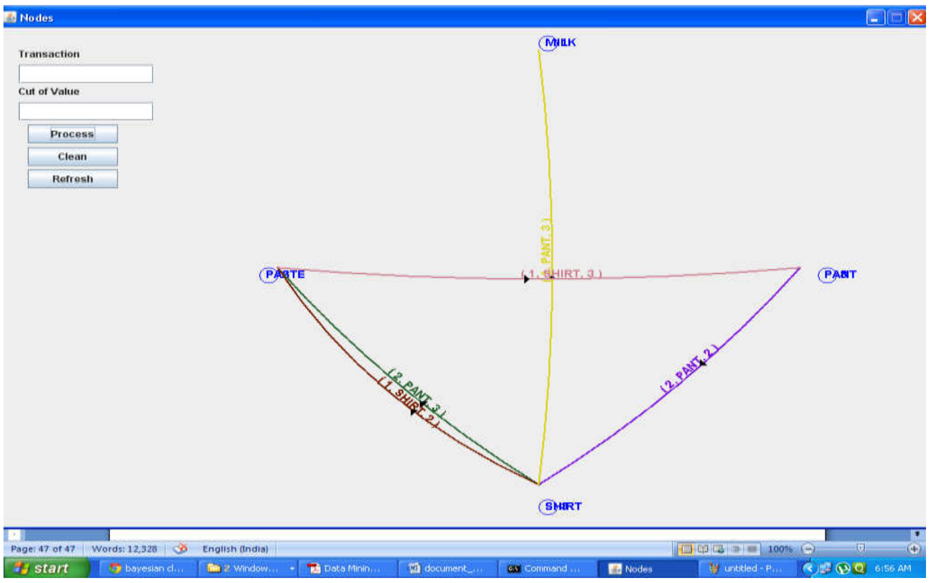


Fig.7.5. Output screenshots

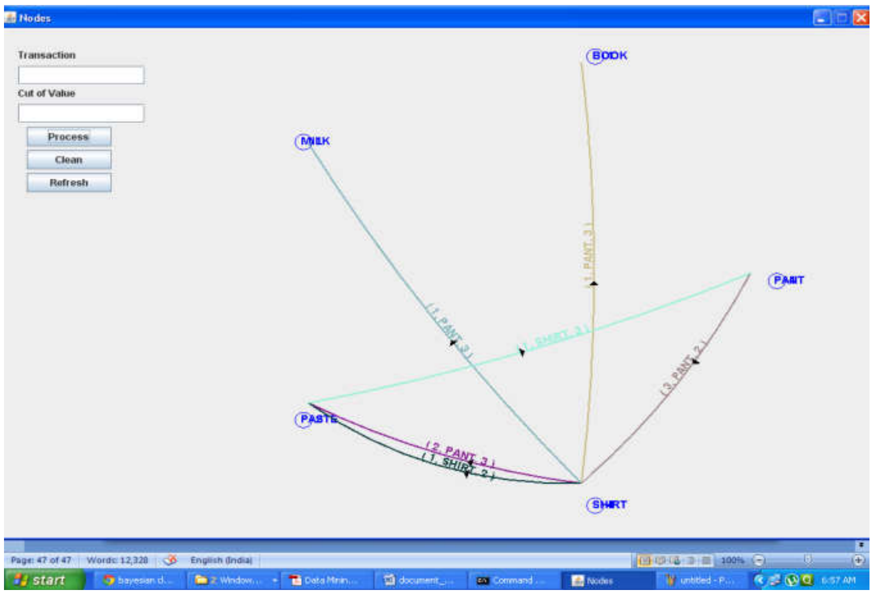


Fig.7.6. Output screenshots

In the Fig 7.7 cut of value 3 is given then refresh button is clicked then it will show the exact pattern which is being purchased >= 3.

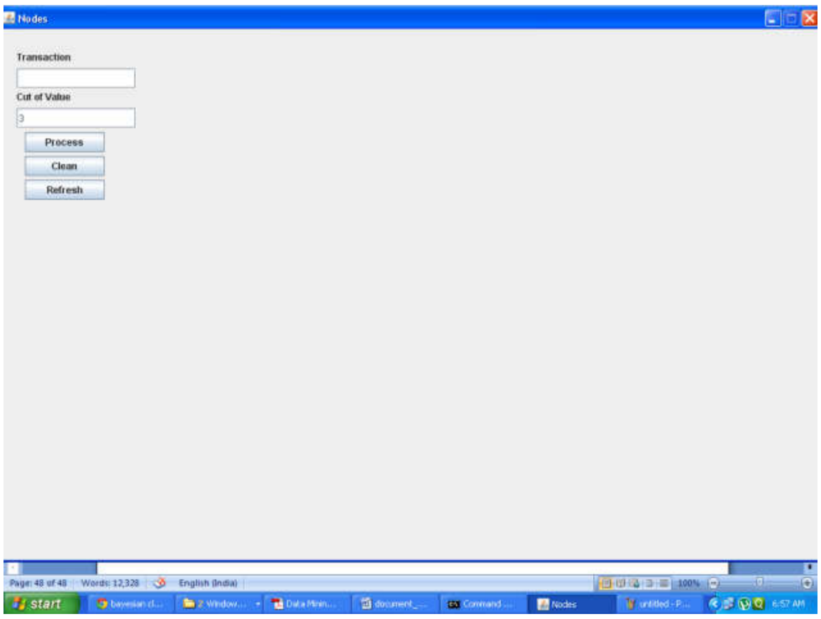


Fig.7.7. Output screenshots

In the previous example we given (pant, shirt) pattern more than 3 times. So it will show the exact pattern as shown below in the Fig 7.7

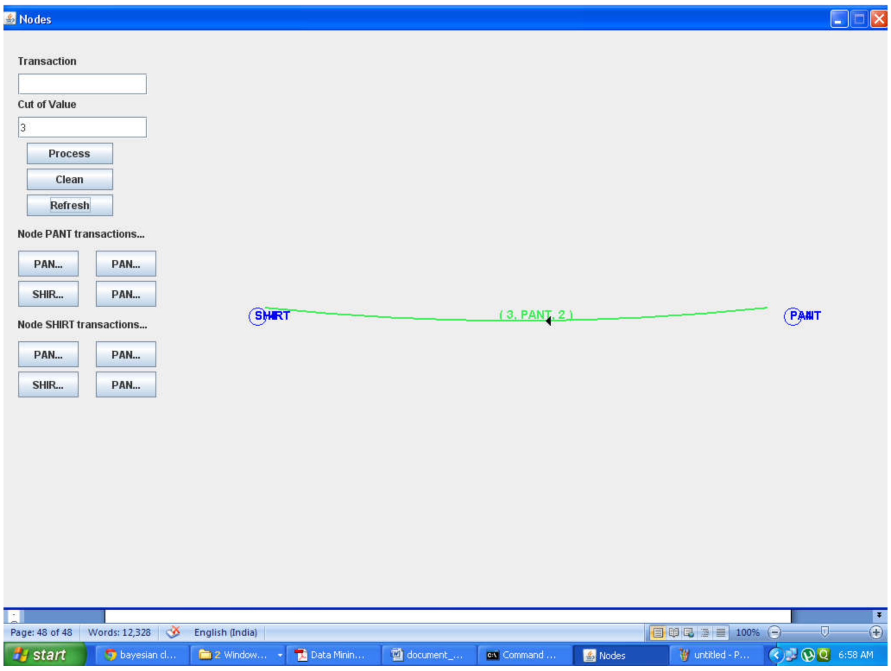


Fig.7.7. Output screenshots

8. Conclusion

In this article, we present a brief overview of the current status and future directions of frequent pattern mining. With over a decade of extensive research, therehave been hundreds of research publications and tremendous research, development and application activities in this domain. It is impossible for us to give acomplete coverage on this topic with limited space and our limited knowledge.Hopefully, this short overview may provide a rough outline of the recent workand give people a general view of the eld. In general, we feel that as a youngresearch eld in data mining, frequent pattern mining has achieved tremendousprogress and claimed a good set of applications. However, in-depth research isstill needed on several critical issues so that the eld may have its long lastingand deep impact in data mining applications.

REFERENCES

Agarwal R.and Srikant R. 1994. Fast algorithm for mining association rules, VLDB, 487-499.

Agarwal, R., Aggarwal, C. and Prasad. V. V. V. 2000. A tree projection algorithm for generation of frequent itemsets. In Journal of Parallel and Distributed Computing (Special Issue on High Performance Data Mining).

Agrawal, R. and Srikant. R. Mining sequential patterns. ICDE'95, 3-14, Taipei, Taiwan.

Agrawal, R. and Srikant. R. Fast algorithms for mining association rules. VLDB'94 487-499, Santiago, Chile.

Agrawal, R., Imielinski, T. and Swami, A. Mining association rules between sets of items in large databases. SIGMOD'93, 207-216, Washington, D.C.

Bayardo. R. J. Efficiently mining long patterns from databases. SIGMOD'98, 85-93, Seattle, Washington.

Brin, S., Motwani, R. and Silverstein. C. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97, 265-276, Tucson, Arizona.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, *Introduction to Algorithms*, MIT Press, Massachusetts: 1998

Gao, Jun: “A New Algorithm of Association Rule Mining”, 2008. International Conference on Computational Intelligence and Security.

Han, J. 2000.“Mining frequent patterns without candidate generation”, In Proc.

Han, J. and Kamber, M. 2006. “Data Mining: Concepts and Techniques”. 2nd Edition, Morgan Kanufmann Publishers, August.

http://en.wikipedia.org/wiki/Apriori_algorithm

http://www.google.com

Vaarandi Risto, “A Breadth- First Algorithm for Mining Frequent Patterns from event logs”, Department of Computer Engineering, Tallinn University of Technology.

****