2.1 INTRODUCTION

Computer Scientists often refer to Moore's law, which states that processing speed of a computer doubles about every eighteen months. It is less known that computer storage capacity doubles about every nine months. Like an ideal gas, computer databases expand to fill available storage capacity. The resulting large amounts of data in databases represent an untapped resource. Business is inherently competitive and in this competitive world of business, one is constantly on the look out for ways to beat the competitions. A question that naturally arose is whether the enormous data that is generated and stored as archives can be used for improving the efficiency of the business performance.

Again it is difficult to convey the vast amount of unused data stored in very large databases at companies, universities, government facilities, and other institutions throughout the world at its current rate of increase. The Library of Congress is estimated to contain 3 petabytes (3000 terabytes) of information [41]. Lesk [41], estimates that about 160 terabyte of information are produced each year worldwide. And, he estimates that there will be over 100,000 terabytes of disk space
sold. It could soon be the case that computer data storage will exceed human capability to use that storage and the data it contains. A process for converting large amounts of data to knowledge will become invaluable. A process called Knowledge Discovery in Databases (KDD) has evolved over the past ten to fifteen years for this purpose. Data mining algorithms are included in the KDD process.

A typical database user retrieves data from databases using an interface to standard technology such as SQL. A data mining system takes this process a step further, allowing users to discover new knowledge from the data.

Data mining is a process of finding value from volume. In any enterprise, the amount of transactional data generated during its day-to-day operations is massive in volume. Although these transactions record every instance of activity, it is of little use in decision-making. Data mining attempts to extract smaller pieces of valuable information from this massive database.

One important aspect of data mining is that it scans through a large volume of data to discover patterns and correlations between attributes. Thus, though there are techniques like clustering, decision trees, etc., existing in different disciplines, they are not readily applicable to data mining, as they are not designed to handle large amount of data. So, in order to apply statistical and mathematical tools, they are to be modified so that they can be able to handle large volume of data. Thus many researchers state that data mining is not yet a well-ordered discipline. The major opportunities for improvement in data mining technology are scalability
and compatibility with database systems, as well as the usability and accuracy of data mining techniques.

The evolution of data mining began when business data was stored in computers, and technologies were generated to allow users to navigate through data in real time. This evolution is due to the support of three technologies that are sufficiently matured: massive data collection, high performance computing and data mining algorithms.

There are several data mining techniques. Out of these Clustering, Association Rules Mining, Classifications etc. are very popular among database community. We briefly discuss some of the techniques below.

2.2 DATA MINING TECHNIQUES

2.2.1 CLUSTERING

Clustering is a useful technique for the discovery of data distribution and patterns in the underlying data. The goal of clustering is to discover both the dense and sparse regions in a dataset. Data clustering has been studied in Statistics, Machine Learning, and Database Communities with diverse emphasis. The earlier approaches do not adequately consider the fact that the dataset can be too large to fit in the main memory. In particular, they do not recognize that the problem must be viewed in terms of how to work with limited resources. The main emphasis has been to cluster with as high as accuracy possible, while keeping the I/O costs high. There are two main approaches to clustering—hierarchical clustering and
partitioning clustering. The *partition* clustering techniques partition the dataset into a predefined number of clusters. They attempt to determine *k* partitions that optimize a certain criterion function. The partition clustering algorithms are of two types: *k*-mean algorithms and *k*-medoid algorithms.

The *hierarchical* clustering techniques do a sequence of partitions, in which each partition is nested into the next partition in the sequence. It creates a *hierarchy* of clusters from small to big or big to small. The *hierarchical* techniques are of two types—agglomerative and divisive clustering techniques. *Agglomerative* clustering techniques start with as many clusters as there are records, with each cluster having only one record. Then pairs of clusters are successively merged until the number of clusters reduces to *k*. At each stage, the pairs of the clusters that are merged are the ones nearest to each other. If the merging is continued, it terminates in a *hierarchy* of clusters, which is built with just a single cluster containing all the records, at the top of the hierarchy. *Divisive* clustering techniques take the opposite approach from *agglomerative* techniques. This starts with all the records in one cluster and then tries to split that cluster small pieces.

A couple of algorithms were developed to find pattern in the data [4] viz. *k*-mean, *k*-medoid, *CLARA, CLARANS, DBSCAN, BIRCH, CURE, STIRR, ROCK, CACTUS*, etc. Still works are going on to develop new algorithms, which successfully capture patterns from different types of complex datasets.
2.2.2 ASSOCIATION RULE MINING

Among the areas of data mining the problem of deriving patterns in the form of associations from data has received a great deal of attention. The problem was formulated by Agrawal et al [47] in 1993 and is often referred as the market-basket problem. In this problem, given a set of items and a large collections of transaction, which are subsets of the item set called as baskets, the task is find a relationship between presence or absence of items within these baskets. There are a large number of available algorithms such as A-priori algorithm, Partition algorithm, Pincer-search algorithm, Dynamic Item set Counting algorithm, Border algorithm etc. which are used to find association rules among the different items in the datasets. Since association rules mining are subject matter of our thesis, we will discuss it in details in the subsequent sections.

2.2.3 CLASSIFICATION

Classification involves finding rules that partition the data into disjoint groups. The input for the classification is the training dataset, whose class labels are known. Classification analyzes the training dataset and constructs a model based on the class label, and aims to assign a class label to the future unlabelled records. Since the class field is known, this type is known as supervised learning. A set of classification rules are generated by such a classification process, which can be used to classify future data and develop a better understanding of each class in the dataset. There are several classification discovery models. They are: the decision trees, neural networks, genetic algorithms and the statistical models like
linear/geometric discriminates. The applications include the credit card analysis, banking, medical applications etc.

Although Clustering, Association rules, Classification are three main areas of research in data mining algorithms in recent years, a known technique, which attempts to learn from data can in principle, be applied for data mining purposes. But it is crucial that these algorithms should be suitably modified to handle the large data residing in secondary memory. This problem is better illustrated by considering the case of prepositional reasoning of AI. Researchers have been proposing algorithms with increasing efficiency, for computing prime implicates/implicants, converting a CNF to DNF, and so on. It is a trivial exercise to notice that these problems are directly related to finding frequent sets [22]. Except for the theoretical importance, these algorithms are inefficient to handle disk-resident data and unless approximately modified, these are unsuitable in practice for data mining applications. This is just a single case of a variety of known techniques in other disciplines, which share same theoretical foundation as the techniques required for data mining, but cannot be readily applied for practical data mining applications. In general, data mining algorithms aim at minimizing I/O operations of disk-resident data, whereas conventional algorithms are concerned about time and space complexities, accuracy and convergence. A few other techniques hold promise of being suitable for data mining purposes. These are Neural Network (NN), Genetic Algorithms (GA), Rough Set Theory and Support Vector Machines (SVM).
2.2.4 WEB MINING AND TEXT MINING

In recent years we have witnessed an ever-increasing flood of written information, culminating in the advent of massive digital libraries. The World Wide Web has become a very popular medium of publishing. Though the Web is rich with information, gathering and making sense of this data is difficult because publication on the Web is largely unorganized. So the problem is to extract implicit, previously unknown information from the massive collection of documents available in the web. There are three modes of web mining namely web content mining, web structure mining and web usage mining. These three approaches are not independent of each other and any efficient mining of the web would require a judicious combination of information from all three sources.

Clustering of hyper-linked documents can rely on a combination of textual and link-based information. Similarly, information about the web structure would greatly enhance the capacity of web usage mining. Web content mining mostly concentrates on text mining, and the textual content in the web can be in an unstructured, semi-structured or structured form. For an unstructured text, features are extracted to view the document in a structured form. Similarly, for unstructured text such as XML, features can be extracted for mining purposes.

2.3 RECENT WORKS IN ASSOCIATION RULE MINING

We now discuss some of the recent works related to the frequent sets and association rule mining, which is the subject matter of our thesis. In [45], Agrawal and Srikant have discussed fast algorithms for mining association rules in large databases of sales transaction. They have presented two new algorithms for solving this problem that are fundamentally different from earlier known algorithms. They have also shown how the best features of the two proposed algorithms can be combined into a hybrid algorithm, A-priori Hybrid, which is much efficient. In [5], an algorithm is discussed that is fundamentally different from earlier known algorithms in the sense that it reduces both CPU and I/O expenses and the algorithm is especially suitable for large size databases. The algorithm is also ideally suited for parallelization.

Quantitative association rule mining is an extension of association rule mining problem. In [52], Srikant and Agrawal discussed the problem of association rule mining in relational tables containing both quantitative and categorical attributes. Similar works have been done in [34]. The authors have described a technique APACS2 for mining interesting quantitative association rules from very large databases. Their technique does not require the user-supplied thresholds and has the ability to discover both positive and negative association rules.

In [39], Zaki and Hsiao discusses an algorithm called CHARM, which minimizes the frequent set mining in other words it finds out the set of closed frequent item
sets, which is much smaller than the set of all frequent item sets from which association rules can be extracted.

In [25], Toivonen et al have discussed an efficient method of pruning the large rules, which cannot be presented to the user. They have shown ways of pruning the set of rules by forming rule covers. A rule cover is a subset of the original set of rules such that for each row in the relation there is an applicable rule in the original set. They have also discussed grouping of association rules by clustering and presented some experimental results of both pruning and grouping.

Typically association rule mining problem operate in a bottom-up breadth-first search direction. The computation starts from frequent 1-item sets and continues until all maximum frequent item sets are found. During the execution, every frequent item set is explicitly considered. Such algorithms perform well when all maximal frequent item sets are short. However, performance drastically decreases when some of the maximal frequent item sets are relatively long. In [12], Lin and Kedem, have presented a new algorithm which combines both bottom-up and top-down directions. The main search direction is still bottom-up but a restricted search is conducted in the top-down direction. It performs well even when some maximal frequent item sets are long. The algorithm produces the maximum frequent set, i.e., the set containing all maximum frequent item sets, which therefore specify immediately all frequent item sets.

So far, the algorithms discussed, mine association rules, which are positive in the sense that such rules deal with the association between items present in the transactions. However, mining negative rules is also an interesting problem.
Negative association rule considers same item as in case of positive association rules, but in addition it also considers negated items i.e. items absent from transactions. Negative association rules are important in market basket analysis to identify the products that conflict with other or product that complements other.

In [42], Antonie and Zaiane proposed an algorithm that extends the support-confidence framework with a sliding correlation coefficient threshold. In addition to finding confident positive rules that have a strong correlation, the algorithm discovers negative association rules with strong negative correlation between the antecedents and consequents.

The problem of mining association rules over interval data is also an interesting data-mining problem. The interval data are ordered data for which separation between data points has meaning. In [49], Miller and Yang have worked on this problem. The authors have given a new definition of association rule mining that is appropriate for interval data. They have presented an algorithm for finding such rules that adjusts the distance-based quality and rule interest measures based on the available memory. When there exist prolific patterns and long patterns in the datasets the candidate generation is found to be costly. In [28], Jiawei Han et al suggest a method of mining frequent patterns without candidate generation. The authors have proposed a frequent pattern (FP-tree) structure, which is an extended prefix tree structure for storing compressed, crucial information about frequent patterns, and develop an efficient FP-tree, based mining method.

Association rule mining is sometimes costly, as it requires multiple passes on the databases. Richard Relue [50] proposes a method, which requires only one scan
of the database and support update of patterns when new data becomes available. They design a new structure called Pattern Repository (PR), which stores all of the relevant information in a highly compact form and allows direct derivation of the FP-tree and association rules quickly with a minimum resource. In addition, it supports run-time generation of association rules by considering only those patterns that meet on-line requirements.

Pawlak develops rough set theory in 1980's [60]. He introduced an early application of rough set theory to knowledge discovery systems, and suggested that rough set approach can be used to increase the likelihood of correct predictions by identifying and removing redundant variables. After that it was found to be a successful tool for data mining and using this theory, rules similar to maximal association can be found. In [24], H. S. Ngugen and S. H. Nguyen presented a novel approach to generation of association rules, based on Rough Set and Boolean reasoning method. They also showed the relationship between the problems of association rule extraction for transaction and relative reducts or α-reducts generation for a decision table. Moreover, their approach can be used to extract association rule in general form. Similar works were done in [30] and [33]. In [30], the authors have showed that the rough set approach to discovering knowledge is much simpler than the maximal association rule mining method. In [33], the authors have proposed a rough set based process which extracts the most appropriate rules, thus reducing the number rules.
2.4 MINING RULES FOR TEMPORAL DATA

One important problem that arises during the discovery process is treating data with temporal information. The attributes related with the temporal information present in this type of datasets need to be treated differently from other kinds of attributes. Most of the earlier data mining techniques tend to treat temporal data as an unordered collection of events ignoring its temporal information. A couple of approaches of Temporal Data Mining [26] have come out till today and sequence mining, episode discovery, time-series analysis for temporal data mining are most important among them. Since Temporal Data Mining is the main subject matter of our work, we will discuss it more elaborately in section-2.7 of this chapter.

2.5 FUZZY IN ASSOCIATION RULE MINING

The word fuzzy came into existence to describe the uncertain situations. We live in a world where we always face some uncertainty in our day-to-day life. The problems with uncertainty have compelled the thinking mind to look into it. Researchers in the sciences have taken the problem with serious consideration so that we may be able get rid of this problem to some extent. Moreover, uncertainty prevails in every human reasoning, which is imprecise in nature. To describe such uncertainty, which appears in the form imprecision, vagueness and doubtful data, we use the word fuzzy. The concept was originated in the United States of America in the University of California by L.A. Zadeh [36] in the year 1965, is
based on the theory of fuzzy subsets which do not have sharply defined boundaries.

In usual situations, there are only two acceptable possibilities for an element, being a member or not being a member of a subset. On the other hand in case of fuzziness, an element is a member in an uncertain fashion. Such situations are termed as fuzzy situation. Being a member is denoted by membership value 1 and not being a member is denoted by membership value 0. It may so happen that the membership may lie between 0 and 1. Functionally, the value of the membership is expressed with the help of membership function. As such fuzzy sets can be represented by membership functions in which the transition from membership to non-membership is gradual rather than abrupt.

Since 1965, the fuzzy set theory has made inroads in almost every direction and has been applied to many branches of human knowledge viz. Economics, Biology, Geology, Social Science, Computer Science, Management Science etc. In section 2.9, we discuss some basic definitions and notations related to fuzziness.

The concept of fuzzy set has entered into association rule mining by introduction of some research articles [10], [55] and [56]. In [10], the fuzzy association rules were defined in this way, 'If \( X \) is \( A \) then \( Y \) is \( B' \) where \( X, Y \) are set of attributes and \( A, B \) are fuzzy sets which describe \( X \) and \( Y \) respectively. In [55], a technique called \( F-APACS \), for mining fuzzy association rules which is a generalization of discretisation of domains of quantitative attributes. Instead of using intervals, \( F-APACS \) employs linguistic terms to represent the revealed regularities and
exceptions. In [13], a theoretical basis of fuzzy association rules is provided by generalizing the classification of the data stored in a database into positive, negative and irrelevant examples of a rule. Accordingly fuzzy sets were used to find fuzzy temporal association rules [57] and [58]. In these works a data mining system for discovering interesting patterns from large datasets is proposed. The patterns by above techniques are expressed in fuzzy temporal association rules, which satisfy the temporal requirements specified by user. Temporal requirements specified by human being tend to be ill-defined or uncertain. To deal with this kind of uncertainty, fuzzy calendar algebra is developed to allow users to describe desired temporal requirements in fuzzy calendars easily and naturally.

2.6 ALGORITHMS FOR MINING ASSOCIATION RULES

2.6.1 PROBLEM DESCRIPTION

2.6.1.1 DEFINITIONS AND NOTATIONS

Now we review certain basic definitions related to association rule mining:

Let \( I \) be a set of items. A set \( X=\{l_1,l_2, \ldots l_m\} \subseteq I \) is called an item set, or a \( k \)-item set if it contains \( k \) items.
A transaction over $I$ is a couple $t = (tid, A)$ where $tid$ is the transaction identifier and $A$ is an item set. A transaction $t = (tid, A)$ is said to support an item set $X \subseteq I$, if $X \subseteq A$.

A transaction dataset $D$ over $I$ is a set of transaction over $I$.

**DEFINITION 1: COVER**

The cover of an item set $X$ in $D$ consists of the set of transactions identifiers of transactions in $D$ that support:

$$\text{cover}(X, D) = \{tid \mid (tid, A) \in D, X \subseteq A\}$$

**DEFINITION 2: SUPPORT**

A transaction $t$ is said to support an item $l_i$, if $l_i$ is present in $t$. $t$ is said to support a subset of items $X \subseteq A$, if $t$ supports each item $l$ in $X$. An item set $X \subseteq I$ has a support $s$ in $D$, $s(X)_D$, if $s\%$ of transactions in $D$ support $X$.

**DEFINITION 3: FREQUENT SET**

An item set $X \subseteq I$ is said to be a frequent item set in $D$ with respect to $\sigma$, if $s(X)_D \geq \sigma$. Otherwise $X$ will be called infrequent. Or in other words an item set is $X \subseteq I$ is said to be a frequent item set in $D$ with respect to $\sigma$, if its probability of occurring in a transaction in $D$ is greater than or equal to $\sigma$, where $0 \leq \sigma \leq 1$. In symbolically

$$\text{Supp}(X, D) = s(X)_D \geq \sigma$$
DEFINITION 4:

Let $D$ be a *transaction dataset* over a set of items $I$, and minimum support $\sigma$. The collection of frequent item sets in $D$ with respect to $\sigma$ is denoted by

$$F(D, \sigma) = \{X \subseteq I \mid \text{Supp}(X, D) \geq \sigma\}$$

or simply $F$.

The set of frequent sets for a given dataset $D$ with respect to a given $\sigma$, exhibits some interesting properties

I. **Downward Closure Property**: Any subset of a frequent set is a frequent set.

II. **Upward Closure Property**: Any superset of an infrequent set is an infrequent set.

DEFINITION 5: **MAXIMAL FREQUENT SET**

A frequent set is a *maximal frequent* set if it is a frequent set and no superset of this is a frequent set.

DEFINITION 6: **BORDER SET**

An item set is a *border* set if it is not a frequent set, but all its proper subsets are frequent sets.
**DEFINITION 7: ASSOCIATION RULE**

An association rule is an expression of the form $X \Rightarrow Y$, where $X$ and $Y$ are item sets, and $X \cap Y = \emptyset$. Such a rule expresses the association that if a transaction contains all items in $X$, then that transaction also contains all items in $Y$. $X$ is called *body* or *antecedent*, and $Y$ is called the *head* or *consequent* of the rule.

**DEFINITION 8: SUPPORT OF AN ASSOCIATION RULE**

The support of an association rule $X \Rightarrow Y$ in $D$, is the support of $X \cup Y$ in $D$. An association rule is called *frequent* if its support exceeds a given minimum support threshold $\sigma$.

**DEFINITION 9: CONFIDENCE OF AN ASSOCIATION RULE**

The *confidence* or *accuracy* of an association rule of an association rule $X \Rightarrow Y$ in $D$ is the conditional probability of having $Y$ contained in a transaction, given that $X$ is contained in that transaction. In symbolically

$$Conf(X \Rightarrow Y) = P(Y|X) = \frac{\text{Supp}(X \cup Y)}{\text{Supp}(X)}$$

An association rule $X \Rightarrow Y$ is said to be confident if $P(Y|X)$ exceeds a given minimum confidence threshold $\tau$. 

24
DEFINITION 10:

Let $D$ be a transaction dataset over a set of items $I$, $\sigma$ a minimum support threshold, and $\tau$ a minimum confidence threshold. The collection of all frequent and confident association rules with respect to $\sigma$ and $\tau$ is denoted by

$$R(D, \sigma, \tau) = \{X \Rightarrow Y \mid X \subseteq I, X \cap Y = \emptyset, \text{ such that } \text{Conf}(X \Rightarrow Y) \geq \tau, \text{ provided } \text{Supp}(X \cup Y) \geq \sigma\}$$

or simply $R$.

2.6.1.2 PHASES OF RULE MINING

The discovery of association rules is the well-studied problem in data mining. The first algorithm proposed to solve the association rule-mining problem was divided into two phases [45]. In the first phase, all frequent item sets are generated. The second phase consists of all frequent and confident association rules. Except few recent works almost all the association rule mining algorithms comply with this phased strategy. Nevertheless, there exists a successful algorithm, called MagnumOpus that uses another strategy to immediately generate a large subset of all association rules [18]. We will not discuss this algorithm here, as main focus of our study is on frequent item set mining of which association rules are natural extensions. Thus the problem of mining association rule can be decomposed into two subproblems:

SUBPROBLEM 1: (ITEMSET MINING PROBLEM)

Find all sets of items, whose support is greater than the user-specified minimum support threshold, $\sigma$ i.e. to find $F(X, D)$. 
SUBPROBLEM 2: (RULE GENERATION PROBLEM)

Use the frequent item sets to generate the desired rules i.e. to find \( R(D, \sigma, \tau) \).

Note that the Item Set Mining problem is actually a special case of the Rule Generation problem. Indeed, if we are given the support and confidence thresholds \( \sigma \) and \( \tau \), then every frequent item set \( X \) also represents the trivial rule \( X \Rightarrow \phi \), which holds with 100% confidence. Obviously, the support of the rule equals the support of \( X \). Also note that for every frequent item set \( A \), all rules \( X \Rightarrow Y \), with \( X \cup Y = A \), hold with at least \( \sigma \) confidence. Hence, the minimum confidence threshold must be higher than the minimum frequency threshold to be any effect.

Next to the support and confidence measures, a lot of other interestingness measures have been proposed in order to get better or more interesting association rules. Recently, Tan et al [44] presented an overview of various measures proposed in Statistics, Machine Learning and Data Mining literature. In this thesis, we will consider the algorithms within the support-confidence framework as our work is basically based on this framework.

2.6.1.3 A-PRIORI ALGORITHM

It is also called the level-wise algorithm. It was proposed by Agrawal and Srikant [45] in 1994. It makes use of the downward closure property. As the name suggests, the algorithm is a bottom up search, moving upward level-wise in a lattice. However, the nicety of the method is that before reading the dataset at every level, it graciously prunes many of the sets, which are certainly infrequent.
The first pass of the algorithm simply counts item occurrence to determine the frequent 1-itemsets. A subsequent pass, say pass $k$, consists of two phases. First, the frequent item sets $L_{k,1}$ found in the $(k-1)$th pass are used to generate the candidate item sets $C_k$, using the \textit{apriori-gen} function described in Section-2.6.1.3.1. Next, the dataset is scanned and the support of candidates in $C_k$ is counted. For fast counting, the candidates in $C_k$ contained in a given transaction are needed to be determined efficiently. In section-2.6.1.3.3 we describe the subset function used for this purpose. The set of candidate item sets is subjected to a pruning process to ensure that all the subsets of the candidate sets are already known to be frequent item sets. The pruning process is discussed in section-2.6.1.3.2.

The \textit{A-priori} frequent item set discovery algorithm uses the two functions, \textit{candidate generation} and \textit{pruning} at the every iteration. It moves upward in a lattice starting from level-1 till level-$k$, where no candidate set remains after pruning. The algorithm is described below:

\textbf{A-priori Algorithm}

\begin{verbatim}
Input: D, $\sigma$
Output: $F(D, \sigma)$
Initialise: $k:=1$, $C_1:=\text{all the 1-itemsets};$
read the dataset to count the support of $C_1$ to determine $L_1$.
$L_1:=[\text{frequent 1-itemsets}];$
for($K:=2; L_{k-1}<>\emptyset; k++) // k represents the pass number//
do begin
    $C_k:=\text{apriori-gen}(L_{k-1});$ // New candidates
    prune($C_k$)
    for all transaction $t \in D$ do begin
        $C_t:=\text{subset}(C_k, t);$ // Candidates contained in $t$
        for all transaction $c \in C_t$ do
            $c.count++;$
    end
    $L_k:=(c \in C_k / c.count \geq \sigma);$ 
end
$F(D, \sigma):= \bigcup_k L_k$
\end{verbatim}

27
2.6.1.3.1 A-PRIORI CANDIDATE GENERATION

The apriori-gen function takes as argument $L_{k-1}$, the set of all frequent (k-1) item sets, and returns a superset of the set of all frequent k-item sets. The intuition behind the A-priori candidate-generation procedure is that if an item set $X$ has minimum support, so do all subsets of $X$. The function works as follows. First, in the join step, $L_{k-1}$ is joined with $L_k$:

```sql
insert into C_k
select p.item1, p.item2, ... p.item_k-1, q.item_k-1
from L_k-1 with L_k-1
where p.item1=q.item1, ... p.item_k-1=q.item_k-1, p.item_k<q.item_k-1;
```

2.6.1.3.2 PRUNING

The pruning step eliminates the extension of (k-1)-item sets, which are not found to be frequent, from being considered for counting support. The pruning algorithm is described below:

```plaintext
prune(C_k)
for all c in C_k
for all (k-1)-subsets d of c do
  if (d in L_{k-1})
    then C_k=C_k\{c
```

2.6.1.3.3 SUBSET FUNCTION

Candidate item sets $C_k$ are stored in a hash-tree. A node of the hash-tree either contains a list of item sets (a leaf node) or a hash table (an interior node). In an interior node, each bucket of the hash table points to another node. The root of the hash-tree is defined to be at depth 1. An interior node at depth $d$ points to
nodes at depth $d+1$. Item sets are stored in the leaves. An item set is added in this way, starting from the root and going down to tree until a leaf is reached. An interior node at depth $d$, which branch is to be followed is decided by applying a hash function to the $d$th item of the item set. All nodes are initially created as leaf nodes. When the number of item sets in a leaf exceeds a specified threshold, the leaf node is converted to an interior node.

Starting from the root node, the subset function finds all the candidates contained in a transaction $t$ as follows: If we are at a leaf, we find which of the item sets in the leaf are contained in $t$ and add references to them to the answer set. If we are an interior node and we have reached it by hashing the item $i$, we hash on each item that comes after $i$ in $t$ and recursively apply this procedure to the node in the corresponding bucket. For the root node, we hash on every item in $t$.

To see why the subset function returns the desired set of references, consider what happens at the root node. For any item set $c$ contained in transaction $t$, the first item of $c$ must be in $t$. At the root, by hashing on every item in $t$, we ensure that we only ignore item sets that start with an item not in $t$. Similar arguments apply at lower depths. The only additional factor is that, since the items in any item set are ordered, if we reach the current node by hashing the item $i$, we only need to consider the items in $t$ that occur after $i$.

Another data structure that is commonly used is a trie or prefix-tree [[1], [7], [53], [48]]. In a trie, every $k$-item set has a node associated with it, as does its $(k-1)$-prefix. The empty item set is the root node. All the 1-item sets are attached to the root node, and their branches are labeled by the item they represent. Every
other \( k \)-item set is attached to its \((k-1)\)-prefix. Every node stores the last item in the item set it represents, its support, and its branches. The branches of a node can be implemented using several data structures such as a hash table, a binary search tree or a vector. In this thesis, we used trie-data structure.

As we mentioned earlier that lots of algorithm were developed in last decade to deal with the market problems, but still the \textit{A-priori} algorithm is one of the most popular one.

After extracting all frequent item sets next problem is to extract all frequent and confident association rules from the frequent item sets. The algorithm is similar to the \textit{A-priori} frequent item set mining algorithm. We will not discuss algorithm because our main focus is on item set mining.

### 2.7 TEMPORAL DATA MINING AND RECENT WORKS

#### 2.7.1 TEMPORAL DATA MINING

Many applications maintain temporal features that cannot be treated as any other attributes and need special attention. Mining such data is an interesting Data Mining Problem. Considering the time aspect of data in mining technique, we gain some insight into the temporal arrangement of events and thus an ability to discover cause and effect. Temporal data mining is an important extension of data mining and it can be defined as the non-trivial extraction of implicit, potentially useful and previously unrecorded information with an implicit or explicit temporal content, from large quantities of data. It has the capability to infer causal

30
and temporal proximity relationships, and this is something that traditional non-temporal data mining cannot do. It may be noted that data mining from temporal data is not temporal data mining, if the temporal component is either ignored or treated as a simple numerical attribute. Also note that temporal patterns cannot be mined from a dataset, which is free of temporal components by non-temporal data mining techniques. Thus for Temporal Data Mining, the underlying dataset must be a temporal one and specific temporal data mining techniques are also necessary. In [11], Claudia M. Antunes and Arlindo L. Oliveira presented an overview of the techniques proposed that deal specifically with temporal data mining. We discuss some of the temporal data mining techniques below:

2.7.2 RECENT WORKS ON TEMPORAL DATA

2.7.2.1 SEQUENCE MINING

In 1995, Rakesh Agrawal and Ramkrishnan Srikant [46] put forward the method of mining sequential pattern. Given a large set of customer transactions, where each transaction consists of customer-id, transaction time and the item bought in the transaction. They introduce the problem of mining sequential pattern over such databases. They present three algorithms to solve the problem and empirically evaluate their performance using synthetic data. Similar works were done in [51]. There given a database of sequences, where each is a list of transactions ordered by transaction time and each transaction is a set of items, the problem is to discover all sequential patterns with a user-specified minimum support, where the support of a pattern is the number of data sequences that
contain the pattern. In [38], authors present, an efficient algorithm for mining frequent sequences considering a variety of syntactic constraints in the form of length or width limitations on the sequences, minimum or maximum gap constraints, on the consecutive sequence elements, applying a window on allowable sequences incorporating item constraints and finding sequences predictive of one or more classes, even rare ones.

The most general form of the sequence-mining problem [Zaki, 1998 [37]] can be stated as follows:

Let $\Sigma = \{i_1, i_2, \ldots, i_m\}$ be a set of $m$ distinct items comprising the alphabet.

An event is a non-empty, disordered collection of items. The items in an event are in some predefined order. An event is denoted as $(i_1, i_2, \ldots, i_k)$, where $i_j$ is an item in $\Sigma$.

Any event that is given as input is called a transaction. Thus, transactions and events have same structure, except that a transaction is known prior to the process and an event is generated during the algorithm. In below we review some definitions related sequence mining.

**DEFINITION 1: SEQUENCE**

A sequence is an ordered list of events. A sequence $s$ is denoted as $(\alpha_1 \rightarrow \alpha_2 \rightarrow \ldots \rightarrow \alpha_k)$, where $\alpha_i$ is an event. A sequence is called $k$-sequence, if the sum of the cardinalities $\alpha_i$ is $k$. 
A subsequence is a sequence within a sequence, preserving the order. In the other words, its items need not be adjacent in time but their ordering in a sequence should not violate the time ordering of the supporting events. A subsequence can be obtained from a sequence by deleting some items and/or events.

A sequence $s'$ is said to support another sequence $s$ if $s$ is a subsequence of $s'$.

Let $D$ be the dataset of input sequences and a sequence is a set of temporally ordered transactions.

**DEFINITION 2: FREQUENCY**

The frequency of a sequence $s$ with respect to this dataset $D$, is the total number of input sequences in $D$ that support it.

**DEFINITION 3: SEQUENCE**

A frequent sequence is a sequence whose frequency exceeds some user-specified threshold. A frequent sequence is said to be maximal if it is not a subsequence of another frequent sequence.

The rationale behind frequent sequences lies in detecting precedence and causal relationships that make them statistically remarkable.

A couple of algorithms were already developed namely, The GSP algorithm, SPADE, SPIRIT etc.
2.7.2.2 CYCLIC ASSOCIATION RULES

If the association rules are computed over monthly sales data, some seasonal variation may be observed where certain rules are true at approximately in the same month of each year. Similarly, association rules can also display regular hourly, daily, weekly, etc. In [6], B. Ozden et al. studied the problem of discovering association rules that display regular cyclic variation over time. They demonstrate that the existing methods cannot be plainly extended to solve this problem of cyclic association rules. They presented two algorithms for discovering such rules. The first one they call sequential algorithm, treats association rules and cycles more or less independently. By studying the interaction between association rules and time, they devise a new technique called cycle pruning, which reduces amount of time needed to find cyclic association rules. The second algorithm, which they call the interleaved algorithm, uses cycle pruning and other optimization techniques for discovering cyclic association rules. They have shown experimentally that interleaved algorithm performed better than sequential algorithm.

2.7.2.3 CALENDRIC ASSOCIATION RULES

In [59], the authors studied temporal association rules during time intervals that follow some user-given calendar schemas. An example of such schema is (year, month, day), which yields a set of calendar-based patterns of the form \(<d_3, d_2, d_1>\), where \(d_i\) is either an integer or the symbol *. Such calendar-based patterns represent all daily, monthly, and yearly patterns. For example, \(<2000, *, 16>\) is
such a pattern, which corresponds to time intervals consisting of all the 16th days of all months in year 2000. They define two types of temporal association rules: precise match association rules that require the association rules hold during every interval, and fuzzy match ones that require the association rules hold during most of these intervals. They extended the \textit{A-priori} algorithm, and developed two optimization techniques to take advantage of the special properties of calendar-based patterns. They also studied the performance of the algorithm by using both synthetic and real world datasets. Similar works are done in [20], considering the items introduced into or removed from the database i.e. the items life-span. They present an extended version of the calendar-based algorithm to mine these kinds of association rules.

\textbf{2.7.2.4 TEMPORAL ASSOCIATION RULES}

In [32] Ale et al discussed the technique of mining temporal association rules. We discuss here details of their work because our work reported in the next chapter is based on this work.

Transaction data are temporal. In large volume temporal data, as used for data mining purposes, information may be found related to products that did not necessarily exist throughout the data-gathering period. So some products may be found which, at the moment of performing the mining, have already discontinued. There may be also new products that were introduced after the beginning of the gathering. Some of these new products should participate in the associations, but may not be included in any rule because of support restrictions. However if we
consider just the transactions generated since the product appears in the market, its support might be above the stipulated minimum.

One way to solve this problem is by incorporating time in the model of discovery of association rules. These rules are called Temporal Association Rules.

One sub-product of this idea is the possibility of eliminating outdated rules, according to the user criteria. Moreover, it is possible to delete obsolete item sets as a function of their lifetime, reducing the amount of work to be done in the determination of the frequent item sets and hence, in the determination of the rules.

The temporal association rules introduced by Ale and Rossi [32] are the extension of the non-temporal model. The basic idea was to limit the search for frequent sets of items or item sets to the lifetime of the item set’s members. For that reason, the concept of temporal support was introduced. Thus each rule has an associated timeframe, corresponding to the lifetime of the items participating in the rule. If the extent of a rule’s lifetime exceeds a minimum length stipulated by the user, then analyze if the rule is frequent in that period. This concept allows finding rules that, with the traditional frequency viewpoint, it would not be possible to discover. The method is discussed below.

2.7.2.4.1 THE TEMPORAL MODEL

Let \( T=\{ \ldots, t_0, t_1, t_2, \ldots \} \) be a set of times, countably infinite over which a linear order \( \prec_r \) is defined where \( t_i \prec t_j \) means \( t_i \) occurs before or is earlier than \( t_j \). It is assumed that is isomorphic to \( N \) (natural numbers).
DEFINITION 1:

Let $R=\{A_1, \ldots, A_p\}$ where the $A_i$'s are called items, $D$ is a collection of subsets of $R$ called the transaction database. Each transaction $s$ in $D$ is set of items such that $s \subseteq R$ includes every item of $d$ independently of the moment in which it appears. Associated to $s$ we have a time stamp $t_s$, which represents the valid time transaction $s$.

Every item has a period of life or life-span in the database, which explicitly represents the temporal duration of the item information i.e. the time in which the item is relevant to the user. The life-span of an item $A_i$ is given by an interval $[t_1, t_2]$ with $t_1 < t_2$.

DEFINITION 2:

Let $A_i$ an item of $R$. With each item $A_i$ and database $D$, a life-span defined by a time interval $[A_i, t_1, A_i, t_2]$ or simply $[t_1, t_2]$ is associated if $A_i$ is understood. $I: A_i \rightarrow 2^T$ is a function assigning a life-span to each item $A_i$ in $R$. The life-span is referred as $I_{A_i}$ and the life-span of $d$ as $I_d = \bigcup I_{A_i} \forall i$.

DEFINITION 3:

Let $X \subseteq R$ a set of items, $s$ contains $X$, if $X \subseteq s$. The set of transactions in $D$ that contain $X$ is indicated by $V(X, D) = \{s: s \in D \land X \subseteq s\}$. If the cardinality of $X$ is $k$, $X$ is called a $k$-item set.
The life-span of a \( k \)-item set \( X \), with \( k > 1 \), is \([t, t']\) where \( t = \max\{t_1; t_2\} \) is the life-span of an item \( A_i \) in \( X \) and \( t' = \min\{t_2; \] is the life-span of an item \( A_i \) in \( X \)

As set operations are valid over life-spans, then the life-span \( I_X \) of the \( k \)-item set \( X \), where \( X \) is the union of the \((k-1)\)-item sets \( V \) and \( W \) with life-spans \( I_V \) and \( I_W \) respectively, is given by \( I_X = I_V \cap I_W \).

**DEFINITION 4:**

Let \( X \subseteq R \) be a set of items and \( I_X \) its life-span. If \( D \) is the set of transactions of the database, then \( D_{IX} \) is the subset of transactions of \( D \) whose time-stamps \( t_i \in I_X \). The number of transactions of \( D_{IX} \) is indicated by \( |D_{IX}| \).

In the non-temporal association rule model the following definition of support holds.

**DEFINITION 5:**

The *support* of \( X \) in \( D \), denoted by \( s(X, D) \), is the fraction of the transactions in \( D \) that contains \( X \): \( |V(X, D)| / |D_{IX}| \). The *frequency* of a set \( X \) is its support. Given a *support* threshold \( \sigma \in [0, 1] \). \( X \) is frequent if \( s(X, D) \geq \sigma \). In this case, it is said that \( X \) has minimum support.
DEFINITION 6:

The *support* of \( X \) in \( D \) over its life-span \( I_x \) denoted by \( s(X, I_x, D) \), is the fraction of transactions in \( d \) that contains \( X \) during the interval of time corresponding to \( I_x \):

\[
|V(X, D)| / |D_{IX}|
\]

The *frequency* of a set \( X \) is its support. Given a threshold of support \( \sigma \in [0, 1] \) and threshold of temporal support \( \tau \). \( X \) is *frequent* in its life-span \( I_x \) if \( s(X, I_x, D) \) and \( |I_x| \geq \tau \). In this case, it is said that \( X \) has *minimum support* in \( I_x \).

The *support threshold* or *frequency* \( \sigma \) and *temporal support threshold* \( \tau \) are parameters given by the user and are dependent on the applications.

DEFINITION 7:

A *Temporal Association Rule* for \( D \) is an expression of the form \( X \Rightarrow Y[t_1, t_2] \), where \( X \subseteq R \), \( Y \subseteq R \setminus X \), and is a time frame corresponding to the life-span of \( X \cup Y \) expressed in a granularity determined by the user.

DEFINITION 8:

The *confidence of a rule* \( X \Rightarrow Y[t_1, t_2] \), denoted by \( \text{conf}(X \Rightarrow Y, [t_1, t_2], D) \) is the conditional probability that a transaction of \( D \), randomly selected in the time frame \([t_1, t_2]\) that contains \( X \) also contains \( Y \):

\[
\text{conf}(X \Rightarrow Y, [t_1, t_2], D) = s(X \cup Y, I_x \cup Y, D) / s(X, I_x \cup Y, D)
\]

where \( I_x \cup Y = \{[t_1, t_2]\} \).
DEFINITION 9:

The temporal association rule $X \Rightarrow Y[t_1, t_2]$ holds in $D$ with support $s$, temporal support $|I_{X,Y}|$ and confidence $c$ if $s\%$ of the transactions of $D$ contain $X \cup Y$ and $c\%$ of the transactions of $D$ that contain $X$ also $Y$, in the time frame $[t_1, t_2]$.

Given a set of transactions $D$, and minimum levels of support, temporal support, and confidence, the problem of temporal association rule discovery is to generate all the association rules that have at least given support, temporal support and confidence.

2.7.2.4.2 TEMPORAL RULES DISCOVERY

The discovery of all the temporal association rules in a transaction set $D$ can be made in two phases [45]:

PHASE 1:

Finding of all frequent item sets in their life-span i.e. the item sets whose frequency exceeds the user’s specified minimum support $\sigma$ and temporal support.

PHASE 2:

Using the above frequent sets to find the association rules.
2.7.2.4.2.1 GENERATING FREQUENT ITEM SETS

1. The well-known A-priori algorithm can be modified can be applied for the same purpose taking time into consideration. Here $L_k$ represents the set of frequent $k$-item sets. Each member of this set of frequent will have the following fields: i) item set, ii) lower limits of the lifetime of the item: $t_1$ and $t_2$ (iii) count of support ($F_r$) of the item set in $[t_1, t_2]$ and (iv) total number of transactions ($F_{Tr}$) found in the interval $[t_1, t_2]$. $C_k$ is the set of candidate $k$-item sets. With the above terminology, we have the modified A-priori algorithm as follows:

2. $L_1 = \{1\text{-frequent item sets}\}$; /* here each item set of size-1 is associated the time of its first appearance $t_1$ and the time of its last appearance $t_2$ the $F_r$ and $F_{Tr}$ in $[t_1, t_2]$ */

3. for ($k=2; L_k \neq \emptyset ; k++$) do begin

4. $C_k = \text{apriori-gen}(L_{k-1})$; /* new candidates with their associated lifespan */

5. for each transaction $s \in D$ do begin

6. $C_s = \text{subset}(C_k, s)$; /* candidates c in s and such that timestamp t of s is in the interval $[t_1, t_2]$ of c */

7. for each candidate $c \in C_s$ do $c.F_r ++$;

8. for each candidate $c \in C_k$ do /* such that time-stamp t of s in in the interval $[t_1, t_2]$ of c */

9. update $c.F_{Tr}$;

10. $L_k = \{c \in C_k : (c.F_r \geq \sigma.F_{Tr}) \land (c.t_2 - c.t_1 \geq \tau)\}$

11. end

12. Answer = $\cup_k L_k$;

$L_1$ is obtained in the first pass, in which the items’ occurrences are counted to determine the 1-frequent item sets. For each item set we store its life-span $[t_1, t_2]$. Besides counting the absolute frequency for each item set, $F_r$ the total number of transactions between $t_1$ and $t_2$, $F_{Tr}$. Then if $F_r/F_{Tr} \geq$ minimum support $\sigma$ and if $t_2 - t_1 \geq$ minimum temporal $\tau$, then the item set is said to be frequent.

Every following pass $k$ consists of two phases: in the first pass are obtained the candidate item sets $C_k$ of size-$k$, based on the frequent item sets $L_{k-1}$ of size-$(k-1)$ obtained in the $(k-1)$ pass by means of the function $\text{apriori-gen}$. The life-span of a $k$-item set with $k>1$ is obtained in the following way: if the $k$-item set $u$ is obtained putting together $k$-1-itemset $v$ and $w$, then the life-span of $u$ is $[u.t_1, u.t_2]$ with $u.t_1 = \max \{v.t_1, w.t_1\}$ and $u.t_2 = \min \{v.t_2, w.t_2\}$. In the second phase the transactions of database are read to compute the support of the candidate item sets.
of $C_k$, for which the function subset is used; it determines if each member $c$ of $C_k$ is contained in the transaction $s$. The time-stamp $t$ of $s$ must satisfy $t \in I_c$. The algorithm does as many passes over the database as the maximal cardinality of a candidate item set.

### 2.7.2.4.2.2 GENERATING A-PRIORI CANDIDATES

The candidate generation is made by means of the function apriori-gen. This function takes as $L_{k-1}$, the set of all frequent $(k-1)$-item sets, and returns a superset of the frequent $k$-item sets, each one with their associated life-span represented genericly by the interval $[t_1, t_2]$. This function has been organized into a Join Step and Pruning Step.

1. **Join Step**
   
   In SQL
   
   ```sql
   insert into $C_k$
   select $p$.item$_1$, $p$.item$_2$, ..., $p$.item$_k$, $q$.item$_{k-1}$
   /* each resulting item set will have an associated interval $[t_1, t_2]$ such that
   $t_1$=max($t_1$ of the $(k-1)$-item sets joined), $t_2$=max($t_2$ of the $(k-1)$-item sets joined)
   */
   from $L_{k-1}$p, $L_{k-1}$q
   where $p$.item$_1$= $q$.item$_1$ and ... and $p$.item$_k$= $q$.item$_{k-2}$ and $p$.item$_k$< $q$.item$_{k-1}$
   ```

   In the next step (pruning) all the candidate item sets $c \in C_k$ such that any subset of $c$ with $(k-1)$ items is not in $L_{k-1}$ are deleted. In the same way the item sets $c$ that $|L_c| < \tau$ are deleted too.

2. **Pruning Step:**
   
   for each item set $c \in C_k$ do
   
   if $|L_c| < \tau$ then
       deleted $c$ from $C_k$
   else
      for each $(k-1)$ subsets $s$ in $c$
      if ($s \in L_{k-1}$) then
          delete $c$ from $C_k$
      ```
After finding all the frequent sets, the association rules can be extracted in accordance with the definition 8 of section-2.7.2.4.1 and each rule will have an associated time interval where the rule holds.

2.8 MINING MAXIMAL FREQUENT INTERVALS IN TEMPORAL INTERVAL DATA

Many real life world data are associated with intervals of time i.e. duration of events instead point events. Such datasets are called temporal interval datasets. A record in such data typically consists of the starting time and the ending time (or the length of the time interval) of the transaction, in addition to other fields. A transaction with starting time $s$ and ending time $e$ supports an interval $[a, b]$ only if $s \leq a$ and $b \leq e$ i.e. $[a, b]$ is contained in the interval $[s, e]$. If the number of transactions supporting the interval $[a, b]$ exceeds a predefined threshold, then $[a, b]$ is called a frequent interval. If no frequent intervals contains $[a, b]$ then it will be called as maximal frequent interval. Mining frequent intervals from such datasets allows the users to group the transactions with similar behaviour together. In [31], the author proposed a method of mining maximal frequent intervals from temporal interval data and showed experimentally that their method outperforms previous methods.
2.9 SOME BASIC DEFINITIONS RELATED TO FUZZINESS

Let $E$ be the universe of discourse. A fuzzy set $A$ in $E$ is characterized by a membership function $A(x)$ lying in $[0, 1]$. $A(x)$ for $x \in E$ represents the grade of membership of $x$ in $A$. Thus a fuzzy set $A$ is defined as

$$A = \{(x, A(x)), x \in E\}$$

A fuzzy set $A$ is said to be normal if $A(x) = 1$ for at least one $x \in E$.

An $\alpha$-cut of a fuzzy set is an ordinary set of elements with membership grade greater than or equal to a threshold $\alpha$, $0 \leq \alpha \leq 1$. Thus an $\alpha$-cut $A_\alpha$ of a fuzzy set $A$ is characterized by

$$A_\alpha = \{x \in E; A(x) \geq \alpha\} \text{ [see e.g. [29]]}$$

A fuzzy set is said to be convex if all its $\alpha$-cuts are convex sets.

A fuzzy number is a convex normalized fuzzy set $A$ defined on the real line $R$ such that

1. there exists an $x_0 \in R$ such that $A(x_0) = 1$, and

2. $A(x)$ is piecewise continuous.

Thus a fuzzy number can be thought of as containing the real numbers within some interval to varying degrees.
Fuzzy intervals are special fuzzy numbers satisfying the following.

1. there exists an interval \([a, b] \subseteq \mathbb{R}\) such that \(A(x_0) = 1\) for all \(x_0 \in [a, b]\), and

2. \(A(x)\) is piecewise continuous.

A fuzzy interval can be thought of as a fuzzy number with a flat region. A fuzzy interval \(A\) is denoted by \(A = [a, b, c, d]\) with \(a < b < c < d\) where \(A(a) = A(d) = 0\) and \(A(x) = 1\) for all \(x \in [b, c]\). \(A(x)\) for all \(x \in [a, b]\) is known as the left reference function and \(A(x)\) for \(x \in [c, d]\) is known as the right reference function. The left reference function is non-decreasing and the right reference function is non-increasing [see e.g. [19]]. The area of a fuzzy interval is defined as the area bounded by the membership function of the fuzzy interval and the real line [3].

The support of a fuzzy set \(A\) within a universal set \(E\) is the crisp set that contains all the elements of \(E\) that have non-zero membership grades in \(A\) and is denoted by \(S(A)\). Thus

\[
S(A) = \{ x \in E; A(x) > 0 \} \quad [\text{see e.g. [29]}]
\]

The core of a fuzzy set \(A\) within a universal set \(E\) is the crisp set that contains all the elements of \(E\) having membership grades 1 in \(A\).