INTRODUCTION

1. INTRODUCTION

Data mining is an interdisciplinary area focusing upon methodologies for extracting useful knowledge from data. The ongoing rapid growth of both offline and online data due to the Internet and the widespread use of databases have created an immense need for KDD methodologies. Discovering knowledge from large databases is considered challenging because the mass of data stored in these databases expand and change from time to time. In order to obtain useful information from such exploding databases, powerful algorithms are needed to explore the interesting patterns.

Many data mining businesses use a transactional database to gather information about consumers and individuals whose knowledge has to be mined. A transactional database is defined as a database that consists of one or more data-manipulation statements and queries. These transactional databases can mine data and manipulate tremendous amounts of information about an individual's personal lives, habits and transactions (Weikum and Vossen, 2001).

A database transaction comprises a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions (Bernstein and Newcomer, 2009).

A transaction database is a set of records representing transactions, where each record consists of a number of items that occur together in a transaction. The most famous example of transaction data is market basket data, where each transaction corresponds to the set of items bought by a customer during a single visit to a store. Text documents can also be represented as transaction data. In this case, each document is represented by a set of words, which can be considered as transaction. Another example of transaction databases is the web log file, where information about each web page visited by a user is treated as a transaction.

The main concern of database transaction processing systems is the overwhelming size that has to be handled by the knowledge discovery algorithms. The current need of the data mining field is to develop ways of representation which reduces this size, at the same time maintains all the important and relevant data needed to extract the desired knowledge. In other words, techniques which can produce a compact representation of the existing database and later that can be used for extracting patterns is the current need of the business market. This research analyzes algorithms that produce compact databases for knowledge discovery from large transaction databases like market basket database and web log databases. From these compact representations, association rule mining is applied to mine frequent patterns.

1.1. DATA MINING

Data mining is a multidisciplinary field that was mainly introduced to process large databases and discovers useful information that could help in decision-making (Washio *et al.*, 2008). The database used for knowledge discovery contains many types of data, including text, image, video, audio, hypertext, graphics, etc. These compound types of huge volume of data is a challenge to process and has attracted several researchers and academicians to find techniques for efficient extraction of useful patterns and is a process more commonly referred to as Knowledge Discover in Databases (KDD). New techniques and directions are being proposed in the literature every day.

Data mining tools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. The automated, prospective analyses offered by data mining move beyond the analyses of past events provided by retrospective tools typical of decision support systems. Data mining tools can answer business questions that traditionally were time consuming to resolve. They scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations (Hafez, 2008).

Data mining can be applied in many areas including database technology, artificial intelligence, machine learning, neural networks, statistics, pattern recognition, knowledge-based systems, knowledge acquisition, information retrieval, high-performance computing and data visualization. Eventhough, many researchers have probed into the field of data mining, it still has to go a long way for perfection. As the demand of customers grow the need for understanding the data and prediction of future becomes crucial (Jalali *et al.*, 2008b).

Data mining methods can be broadly categorized into two tasks, namely, predictive mining and descriptive mining. Predictive data mining attempts to do predictions based on inference on available data. They use some variable to predict unknown or future values of other variables. Some predictive data mining techniques are clustering, classification, regression, anomaly detection, change/evolution analysis, etc. Descriptive data mining uses existing or historical data to discover human-interpretable patterns that describes the data. Examples of such techniques include sequential pattern discovery, clustering, characterization, association rule discovery, etc.

Data mining can be applied to different type of data repository like text data (both numbers and alphabets), web data (web links, web contents, etc.), multimedia data (images, audio, video, etc.). The huge volumes of data required to be retrieved, processed and store make compression as promising areas to explore.

1.2 WEB MINING

An important extension to data mining is Web mining, which uses data mining algorithms and methodologies on web documents to retrieve useful business information (Kolari and Joshi, 2004). It studies various aspects of a website and extracts patterns and relationships from user behaviour. It focuses on extracting knowledge from web data by taking into consideration the

IP addresses of website visitors, browser logs, cookies and so on. The details of web data are stored in a special file called "Web Log File".

Web usage mining is one of the important categorization of web mining, where data mining techniques are applied to discover patterns and trends in the browsing behavior of website visitors. Web usage mining tries to discover the useful information such as navigation patterns, frequently visited websites from the secondary data derived from the interactions of the users while surfing on the Web. It focuses on the techniques that could predict user behavior while the user interacts with the Web. E-commerce specialists have recently focused on this behavioral data to understand how users decide to buy something (Changa *et al.*, 2006). It is also widely used by educationalists in e-learning environment (Zaiane and Luo, 2010a, 2010b). It is a technique that is widely used in applications which involves personalization (Jalali *et al.*, 2008a), system improvement (Zorrilla *et al.*, 2005), site modification (Spiliopoulou, 2000), business intelligence (Sarwar *et al.*, 2000; Schafer *et al.*, 2000) and usage characterization (Raju *et al.*, 2010).

In Web Usage Mining, web log data plays a vital role and it can be collected from three main sources, as given in Figure 1.1 (Hu *et al.*, 2010). The three sources are server logs, browser logs, proxy logs that are obtained from an organization's database.

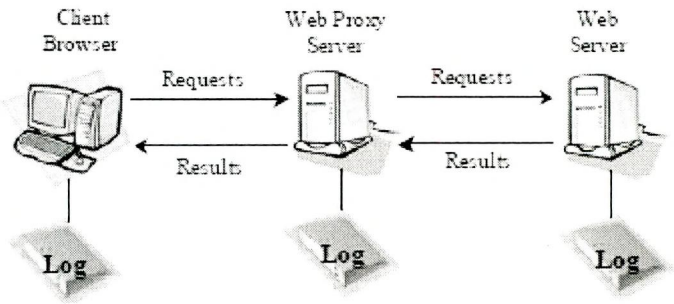

Figure 1.1: Web log file locations

Web Server Logs – These are logs which maintain a history of page requests. The W3C maintains a standard format for web server log files. More recent entries are appended to the end of the file. Information about the request, including client IP address, request date/time, page requested, HTTP code, bytes served, user agent, and referrer are stored in text format. These data can be combined into a single file, or separated into distinct logs, such as an access log, error log, or referrer log. However, server logs typically do not collect user-specific information. These files are not accessible to general Internet users and are available only to the webmaster or other administrative person.

A statistical analysis of the server log reveals traffic patterns by time of day, day of week, referrer, or user agent. Efficient web site administration, adequate hosting resources and the fine tuning of sales efforts can be aided by analysis of the web server logs. Marketing departments of any organization that owns a website should be trained to understand these powerful tools.

Proxy Server Logs - A Web proxy is a caching mechanism which lies between client browsers and Web servers. It helps to reduce the load time of Web pages as well as the network traffic load at the server and client side. Proxy server logs contain the HTTP requests from multiple clients to multiple Web servers. This may serve as a data source to discover the usage pattern of a group of anonymous users, sharing a common proxy server.

Browser Logs – Various browsers like Mozilla, Internet Explorer etc. can be modified or various JavaScript and Java applets can be used to collect client side data. This implementation of client-side data collection requires user cooperation, either in enabling the functionality of the JavaScript and Java applets, or to voluntarily use the modified browser. Client-side collection scores over server-side collection because it reduces the session identification problems. The data thus collected differ in terms of the location of the data source, the kinds of data available, the segment of population from which the data was collected, and methods of implementation.

1.3. TRANSACTION DATABASE

Frequent pattern mining is the discovery of relationships or correlations between items in a transaction database. This research work considers two types of datasets as transaction database to perform frequent pattern mining. They are market basket data and web log data.

1.3.1. Market Basket

A set of market basket transactions is a common dataset used in frequent pattern analysis (Gionis *et al.*, 2007). A dataset is typically in a table format. Each row is a transaction, identified by a transaction identifier or a TID. A transaction contains a set of items bought by a customer. A set of transactions might be organized in either an enumerated (dense), or a sparse binary vector format (Ceglar and Roddick, 2006; Shenoy *et al.*, 2000). In either format a dataset can be processed horizontally or vertically. Figure 1.2 illustrates the data organization formats of a simple market basket dataset.

Figure 1.2a shows the various items along with their item id's while Figure 1.2b shows the transaction database. In Figure 1.2b, each row contains items associated with a customer purchase. This representation is considered simple, as they produce only the purchase information and not other irrelevant details like quantity, profit on items sold, etc.

ItemID	Item
A	Cereals
B	Beer
C	Diaper
D	Egg
E	Milk

(a) Item List

TID	Items
1	{Cereals, Milk}
2	{Beer, Cereal, Diaper, Egg}
3	{Beer, Diaper, Milk}
4	{Beer, Cereal, Diaper, Milk}
5	{Diaper, Milk}

(b) Transaction Database

Figure 1.2 : Example Transaction Database

While using web log data as transactional database, each row is identified as a transaction (Han *et al.*, 2000). From the various fields available, the IP address, time stamp, web pages requested and visited are considered as items. Here, a transaction consists of the various pages visited by a web user.

1.3.2. Web Log Data

A web log file is a text file that is automatically created by the server as records of website activity (Chitraa and Davamani, 2010). The present research work uses an IIS format of log file and is explained below.

An IIS format is a fixed (cannot be customized) ASCII format which includes basic items, such as the user's IP address, user name, request date and time, service status code, and number of bytes received. In addition, IIS format includes detailed items, such as the elapsed time, number of bytes sent, action (for example, a download carried out by a GET command), and target file. The items are separated by commas, making the format easier to read than the other ASCII formats, which use spaces for separators. The time is recorded as local time. Figure 1.3 shows an example of web log fields and a sample web log file is shown in Figure 1.4.

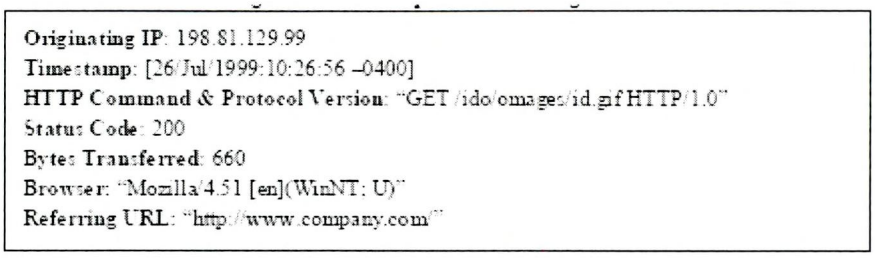

Figure 1.3 : An example of server log files

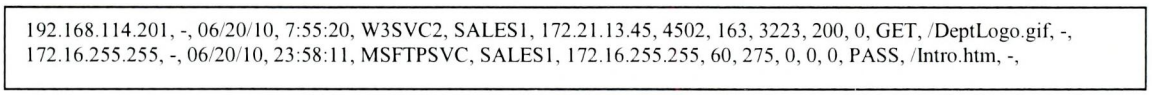

Figure 1.4 : Sample IIS Format Web Log File

In the example, the first entry indicates that an user with the IP address of 192.168.114.201 issued an HTTP GET command for the image file /DeptLogo.gif at 7:55 A.M. on June 20, 2010, from a server named SALES1 with IP address 172.21.13.45. The 163-byte HTTP request had an elapsed processing time of 4502 milliseconds (4.5 seconds) to complete, and returned, without error, 3223 bytes of data to the anonymous user. An empty field is denoted by a hyphen (-) and fields are separated by commas, making the format easier to read than the other ASCII formats, which use spaces for separators. The time is recorded as local time recorded in milliseconds.

The present research work also focus on identifying frequent patterns from web log files, which can be used by web masters and business to improve their websites to suit the needs of their customers. Vast amount of information can be gained from the pattern of user's browsing and association rules are best suited for this purpose (Facca and Lanzi, 2005).

The problem of finding web pages visited together is similar to finding associations among itemsets in transaction databases. Once transactions have been identified, each of them could represent a basket and each resource an item.

A set of sessions is defined as $S = \{S_1, S_2, \ldots S_n\}$ where n is the number of sessions in S and a set of URLs $R = \{url_1, url_2, \ldots url_n\}$ where n is the number of URLs in S. Each session S_i is defined by an IP and URLs :

$$S_i = (IP_i, URLs_i)$$

IP identifies the user of the session and URLS is the set of pages requested by the user. The number of pages requested by a user IP_i in a session S_i is defined as m. A value of 30 minutes is used to define a session. In the present research work, the web log file is termed as the transaction database, each log entry is called as a transaction and the IP address and URLs are the items.

1.4. FREQUENT PATTERN MINING

Since in the beginning of 90s, frequent pattern mining has become one of the most actively researched topic in data mining and knowledge discovery in databases (Aggarwal *et al.*, 2009). This research work analyzes techniques which can compactly represent transaction databases which can then be used for frequent pattern mining. Frequent pattern mining is used by many applications including market basket analysis, click stream analysis, web link analysis, genome analysis and drug design (molecular fragment mining).

Market basket analysis, generally known as frequent itemset mining, describes the shopping behavior of customers of supermarkets, mail-order companies and online shops, for products that are frequently bought together. The name is derived from a person walking through a supermarket throwing all the things to buy in a shopping cart. This "market basket" is then analyzed. The goal here is to discover interesting relationships between retail products in order to help retailers in identifying cross-sale opportunities. For this task, a large number of efficient algorithms were developed, which are based on sophisticated data structures and clever processing schemes. Association rule mining is the most common approach for performing this task. Recent developments tackled the more complex tasks of mining frequent patterns, which have applications in biochemistry, web mining, and citation network and program flow analysis. These tasks are more complicated, since the structure of the data introduces new problems, the solution of which is trivial for item sets.

1.4.1. Basic Notions

The frequent pattern mining problem was first introduced by Agrawal *et al.* (1993) as mining association rules between sets of items. Frequent itemset mining aims at finding regularities in the shopping behavior of customers of supermarkets, mail-order companies, on-line shops etc. More precisely, they find sets of products that are frequently bought together.

Let $A = \{a_1, \ldots, a_m\}$ be a set of items. This set is called the item base. Items may be products, special equipment items, service options, etc. Any subset $I \subseteq A$ is called an item set and an item set may be any set of products that can be bought (together).

Let $T = (t_1, \ldots, t_n)$ with $\forall_i;\ 1 \leq i \leq n : t_i \subseteq A$ be a vector of transactions over A. This vector is called the transaction database. A transaction database can list, for example, the sets of products bought by the customers of a supermarket in a given period of time. Some general properties of a transaction database are

- Every transaction is an item set, but some item sets may not appear in T.
- Transactions need not be pairwise different: it may be $t_i = t_j$ for $i \neq j$.
- T may also be defined as a bag or multiset of transactions.
- The set A may not be explicitly given, but only implicitly as $A = \bigcup_{i=1}^{n} t_i$.

1.5. ASSOCIATION RULE MINING

The first studied approach in the field of frequent pattern mining is based on association rules (Park *et al.*, 1995; Agrawal *et al.*, 1996). Association rules mining was first proposed to find all the rules in the transaction data to analyze the relationship between items purchased by customers in a shop. Association rule mining can be stated as follows.

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items. Let 'D' be a set of transactions, where each transaction 'T' is a set of items such that $T \subseteq I$. An association rule is an implication of the form, $X \rightarrow Y$, where $X \subset I$, $Y \subset I$ and $X \cap Y = \emptyset$. The rule $X \rightarrow Y$ holds in the transaction set T with confidence 'c', if c% of transactions in T that support 'X' also support 'Y'. The rule has support 's' in T if s% of the transactions in T contains $X \cup Y$.

Given a set of transactions *D* (the database), the problem of mining association rules is to discover all association rules that have support and confidence greater than the user-specified minimum support (called *minsup*) and minimum confidence (called *minconf*). An association rule 'r' is a relation between itemsets of the form

$$r: X \Rightarrow (Y - X)$$

Where X and Y are frequent itemsets and $X \subset Y$. The itemsets X and $(Y - X)$ are called, respectively, antecedent and consequence of the rule 'r'. The valid association rules are those for which the measure of support and confidence is greater than or equal to the minimal thresholds of support and confidence, called minsup and minconf (Webb, 2000). Support and confidence are calculated as in Equation (1.1) and (1.2).

$$\text{Support}(X) = \frac{|\{t \in D \mid X \subseteq t\}|}{|D|} \qquad (1.1)$$

$$\text{Confidence}(r) = \frac{\text{Support}(Y - X)}{\text{Support}(X)} \qquad (1.2)$$

A typical association rule is of the form

$$A \Rightarrow B, C \; [\text{ Support} = 60\%, \text{Confidence} = 80\%]$$

"50% of visitors who accessed URLs B and C also visited A"

The process of finding all the association rules with confidence and support above the respective thresholds is a two stage process. The first stage consists in finding all sets of items with support above the support threshold; these sets are called large item sets. The second step consists in computing for each large itemset, the confidence for all its expressions with the form of a rule; the expressions whose confidence is above the confidence threshold are the rules.

The second step of the process is trivial; therefore, the problem of mining association rules can be viewed as the problem of finding all the itemsets with support above a given threshold. The most well known algorithms to perform such task are the Apriori algorithm and FP Growth, which are described by Agrawal *et al.* (1996).

Apriori is the best-known algorithm to mine association rules (Agrawal and Srikant, 1994). It uses a breadth-first search strategy to count the support of itemsets and uses a candidate generation function which exploits the downward closure property of support. The main advantages of apriori algorithm are (i) uses large itemset property (ii) Easily parallelized and (iii) Easy to implement. The major concerns of the algorithm are:

- It is costly to handle a huge number of candidate sets. For example, if there are 10^4 frequent 1-itemsets, the Apriori algorithm will need to generate more than 10^7 length-2 candidates, accumulate and test their occurrence frequencies. Moreover, to discover a frequent pattern of size 100, such as $\{a_1, \ldots, a_{100}\}$, it must generate $2^{100} - 2 \approx 10^{30}$ candidates in total. This is the inherent cost of candidate generation, no matter what implementation technique is applied.

- It is tedious to repeatedly scan the database and check a large set of candidates by pattern matching, which is especially true for mining long patterns.

In order to overcome the drawback inherited in Apriori, J.Han developed an efficient FP-tree based mining method, FP-growth. FP-growth (frequent pattern growth) uses an extended prefix-tree (FP-tree) structure to store the database in a compressed form (Han *et al.*, 2004). FP-growth adopts a divide-and-conquer approach to decompose both the mining tasks and the databases. It uses a pattern fragment growth method to avoid the costly process of candidate generation and testing used by Apriori.

One weakness of FP-Tree is the overhead of constructing many conditional FP-Trees during mining that reduces its performance as the patterns get longer and/or the support level gets lower. In order to address these drawbacks, the present research work analyzes the applicability of two techniques that produces a compact version of original database. The first technique was proposed by Wan and An (2005) and the second technique was proposed by Gopalan and Sucahyo (2004).

1.6. DATA COMPRESSION AND FREQUENT PATTERN MINING

Data compression is an effective method for reducing storage space and saving network bandwidth. A large number of compression schemes have been developed based on character encoding or on detection of repetitive strings, and comprehensive surveys of compression methods and schemes are given in Bell *et al.* (2009), Storer (2005) Lelewer and Hirschberg (2003).

There are two fundamentally different types of data compression: lossless and lossy. A lossy compression technique removes unwanted data which eventhough it degrades the output quality but maintains overall important information. Lossy compression technique, on the other hand, attempts to compress data while retaining all information. Algorithms which are used to reduce the size of transaction databases aim for a lossless compression, which means that the frequent patterns generated by the original transaction database and its corresponding compact version should be identical, with the same input parameters. The main aim here is to reduce the I/O time when mining patterns from a transaction database.

Mining frequent patterns is a fundamental step in data mining and considerable research effort has been devoted to this problem since its initial formulation. A number of data compression strategies and data structures, such as prefix-tree (or trie) (Agarwal *et al.*, 2000; Bayardo, 1998; Brin *et al.*, 1997) and FP-tree (Han *et al.*, 2000), have been devised to optimize the candidate generation and the support counting process in frequent patterns mining. The

concept of prefix-tree is based on the set enumeration tree framework (Rymon, 1992) to enable itemsets to be located quickly. Figure 1.5 illustrates the prefix-tree for the example transaction database in Table 1.1.

Table 1.1

Example Transaction Database

TID	List of itemIDs
001	A, B, C, D
002	A, B, C
003	A, B, D
004	B, C, D
005	C, D
006	A, B, C
007	A, B, C
008	B, C
009	B, C, D
010	C, D

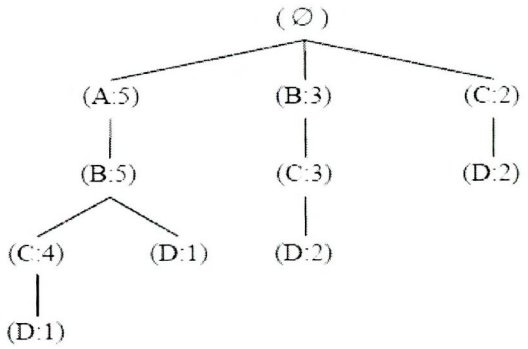

Figure 1.5 : Prefix-tree for the Database in Table I

The root node of the tree corresponds to the empty itemset. Each other node in the tree represents an itemset consisting of the node element and all the elements on nodes in the path (prefix) from the root. For example, the path (∅)-(B:3)-(C:3)-(D:3) in Figure 1.5 represents the itemset {B, C, D} with support of 3.

It can be seen that the set of paths from the root to the different nodes of the tree represent all possible subsets of items that could be present in any transaction. Compression is achieved by building the tree in such a way that if an itemset shares a prefix with an itemset already in the tree, the new itemset will share a prefix of the branch representing that itemset. Further compression can also be achieved by storing only frequent items in the tree. The FP-growth method uses another compact data structure, FP-tree (Frequent Pattern tree), to represent the conditional databases.

FP-tree is a combination of prefixtree structure and node-links, as shown in Figure 1.6.

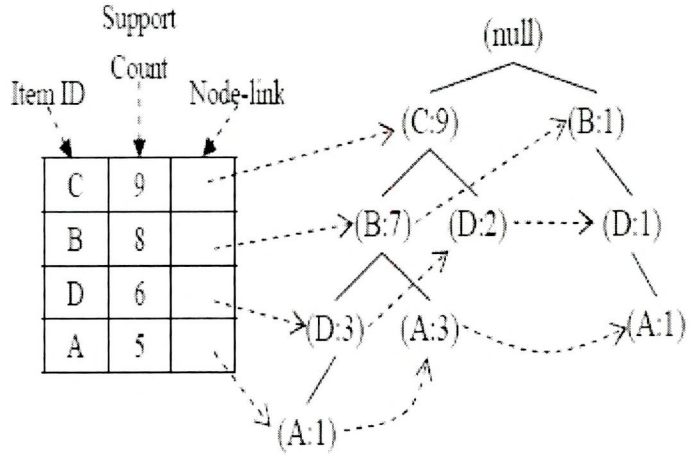

Figure 1.6: FP-Tree for the database in Table 1.1

All frequent items and their support counts are found by the first scan of database, and are then inserted into the header table of FP-tree in frequency descending order. To facilitate tree traversal, the entry for an item in the header table also contains the head of a list that links all the corresponding nodes of the FP-tree. In the next scan, the set of sorted (frequency descending order) frequent items in each transaction are inserted into a prefix-tree as a branch. The root node of the tree is labeled with "null", every other node in the FP-tree additionally stores a counter which keeps track of the number of itemsets that share that node. When the frequent items are sorted in their frequency descending order, there are better chances that more prefixes can be shared, thus the FP-tree representation of the database can be kept as small as possible.

Both the algorithms selected in the present research work, focuses on optimizing the existing algorithms by generating a compact version of the original database as a tree and store it in secondary storage rather than main memory, which can then be used for efficient frequent pattern mining and other mining process and thus can save mining time.

In the approach proposed by Wan and An (2005), in addition, the counter associated with each node in the prefix-tree and FP-tree stores the number of transaction containing the itemset represented by the path from the root to the node. However, each path from every node of the CT-tree to the root represents a unique transaction, and the associated counter records the number of occurrences of this transaction in the original transaction database. Moreover, for a given transaction database, the number of nodes and node-links in FP-tree will change with different minimum support threshold specified by the user. But there is only one unchanged CT-tree for every transaction database. And the FP-tree structure can be constructed from a compact transaction database more efficiently in only one database scan, since the head part of a compact transaction database lists all items in frequency descending order, and the body part stores all ordered transactions associated with their occurrence counts.

Gopalan and Sucahyo (2004) presented a variation to FP-Tree of FP-growth algorithm, called CT-PRO, which divided the database into several projections and then mines each projection independently. The projections are also represented as CFP-Trees. Unlike FP-Growth, the mining process is performed by a non-recursive function and so the overhead of creating an extra data structure at each mining step was avoided.

1.7. MOTIVATION

Due to widespread computerization and affordable storage facilities, there is an enormous wealth of information embedded in huge database belonging to different enterprises. In the domain of scientific computing, the major problem is to infer valuable information from observed data. The key idea of data mining is to find effective ways to combine the computer's power to process data and detect useful patterns.

Businesses today require information about consumer behaviour in a fast manner and they invest hugely by employing specialists for discovering such

patterns. But often, even with the availability of fast and cheaper hardware, this process is very slow. This has introduced the clear need for a technique that can help to harness information in a fast manner and make it available to enterprise applications.

Due to the information explosion and the increase in number of persons using automated internet, Web users are facing the problems of information overload. As a consequence, presenting web users with relevant information has become a critical issue in web-based information retrieval and web based applications.

Today, business strives on customer satisfaction and association rule mining is used by many applications for this purpose. Discovering frequent patterns is considered as a vital step in many marketing sectors to improve their business or to improve their website. The number of rules produced by such algorithms is very huge. Storing these details in hard disk prove to be difficult even with the higher end storage disks. Moreover, the process is time consuming, as a number of scans have to be performed during the process of frequent pattern discovery and rule generation.

Thus, the current need of the market is to have a frequent pattern mining algorithm which can

(i) Reduce the I/O scan and time.
(ii) Reduce the amount of storage used by the transaction database.

The motivation behind building compact database transaction comes from the following points.

(i) A number of transactions in a transaction database may contain the same set of items. For example, as shown in Table 1.1, transaction {A, B, C} occurs thrice, and transactions {B, C, D} and {C, D} both occur twice in the same database. Therefore, if the transactions that have the same set of items can be stored in a single transaction with

their number of occurrence, it is possible to avoid repeated scanning of the same transaction in the original database.

(ii) If the frequency count of each item in the given transaction database can be acquired when constructing the compact database before mining takes place, it is possible to avoid the first scan of database to identify the set of frequent items as most approaches to efficient mining of frequent patterns do.

This research work motivated by these observations, analyzes algorithms which provide a solution to these problems by using a compressed variation of the transactional database.

1.8. PROBLEM STATEMENT AND OBJECTIVES

The problem statement for the present research work is formulated as below:

> *"Given a transaction database* $T = <t_1, t_2, ..., t_n>$ *consisting of a sequence of transactions T, itemset* T_A *consisting of a set of items* (t_i, I)*, support and confidence value, find a compact representation of T, T', such that, the size of T is less than T' which facilitates the finding of all frequent itemsets from T' and generate association rules with a minimum confidence."*

To develop a frequent pattern mining system that satisfies the above problem statement, an architecture proposed by Wan and An (2005) and Gopalan and Sucahyo (2004), hereafter referred as CT-Apriori and CT-PRO respectively, are selected. The following objectives were formulated.

1. To produce a condensed version of a transactional database to reduce storage space, I/O-time spent on one database scan.

2. To implement algorithms, CT-Apriori, for compact transaction database for efficient frequent pattern mining.
3. To implement a compressed form of FP-Tree for FP-Growth algorithm for efficient frequent pattern mining.
4. To propose a preprocessing module for converting a web log data into a format suitable for transaction frequent pattern mining.
5. To analyze the applicability of CT-Apriori and CT-PRO algorithms on mining frequent patterns from web log data.
6. To compare the performance of CT-Apriori and CT-PRO algorithms in terms of number of association rules generated, storage space used and time taken to perform frequent pattern mining.

1.9. LAYOUT OF THE DISSERTATION

The underlying objective of this research work is to compare algorithms that create compact transaction databases and study the effect of these databases on frequent pattern mining. Two type of transaction databases are taken into account. The first is a market basket database and the second is the web log data. Thus the applicability of compact database representation for web log mining is also studied in this research work. The dissertation is organized as follows.

Chapter 1 provides a brief introduction to frequent pattern mining with special focus on associative rule mining. The objectives of the research work are outlined here.

The literature review (chapter 2) provides a critical look at the existing research in the area. Several researchers have addressed the problem of frequent pattern mining.

The main component of the selected algorithm is the creation of a compact representation of the database which will facilitate easy frequent pattern mining. Two algorithms based on Apriori and FP-Growth was selected

and the methodology of these algorithms is discussed in Chapter 3, along with the preprocessing steps used while using web log data as transaction database.

To analyze the performance of the two proposed compact transaction frequent pattern mining, several experiments were conducted and the results obtained are tabulated and discussed in Chapter 4. The conclusion of the research work is summarized along with future research directions in Chapter 5. The work of several researchers are quoted and used as evidence to support the concepts explained in this dissertation. All such evidences used are listed in the reference section of the dissertation.

1.10. CHAPTER SUMMARY

This chapter provided a brief introduction to data mining, web mining and frequent pattern mining. The objectives formulated were also outlined and is presented in the next chapter, Review of Literature.