`

A
Dissertation
On

"Improving Performance in Mining Algorithms"

Submitted in Partial fulfillment of the requirement
For the award of Degree of

MASTER OF TECHNOLOGY
IN
Computer Technology and Applications
Delhi Technological University, Delhi

SUBMITTED BY

Abhinav Kumar Dwivedi
University Roll No: 02/CTA/2K10

UNDER THE GUIDANCE OF

Mrs. Malaya Dutta Borah

Assistant Professor
Delhi Technological University

DEPARTMENT OF COMPUTER ENGINEERING
DELHI TECHOLOGICAL UNIVERSITY
2010-2012

`

CERTIFICATE

This is to certify that the work contained in this dissertation entitled "**Improving Performance in Mining Algorithms**" submitted in the partial fulfillment, for the award for the degree of M.Tech in Computer Technology and Applications at **DELHI TECHNOLOGICAL UNIVERSITY** by **ABHINAV KUMAR DWIVEDI, Roll No. 02/CTA/2K10** is carried out by him under my supervision. This matter embodied in this project work has not been submitted earlier for the award of any degree or diploma in any university/institution to the best of our knowledge and belief.

(Mrs. Malaya Dutta Borah)
Project Guide
Assistant Professor
Department of Computer Engineering
Delhi Technological University

`

ACKNOWLEDGEMENT

First of all, let me thank the almighty god and my parents who are the most graceful and merciful for their blessing that contributed to the successful completion of this project.

I feel privileged to offer sincere thanks and deep sense of gratitude to **Mrs. Malaya Dutta Borah**, project guide for expressing her confidence in me by letting me work on a project of this magnitude and using the latest technologies and providing their support, help & encouragement in implementing this project.

I would like to take this opportunity to express the profound sense of gratitude and respect to all those who helped us throughout the duration of this project. DELHI TECHNOLOGICAL UNIVERSITY, DELHI, in particular has been the source of inspiration, I acknowledge the effort of those who have contributed significantly to this project.

ABHINAV KUMAR DWIVEDI
(University Roll No.: 02/CTA/2K10)

`

ABSTRACT

In Frequent pattern mining, we have to generate and examine a large number of intermediate subsequences. So it's a difficult data mining problem with broad applications. Frequent Pattern Mining has been at the core of the field of data mining for the past many years. Frequent Pattern Mining is the task of finding sets of items that frequently occur together in a dataset. With the higher growth of data, users need more relevant and sophisticated information, which may be lying hidden in the data. Data mining is often described as a discipline to find hidden information in a database. It involves different algorithms and techniques to discover useful knowledge lying hidden in the data.

Most of the previously developed sequential pattern mining methods, such as *Generalized Sequential Pattern* algorithm (GSP), explore a candidate generation-and-test approach [1] to reduce the number of candidates to be examined. However, this algorithm may not be efficient in mining sequence databases which contains a number of numerous patterns and/or long patterns. Since it produces a large number of candidate sequences and does a lot scan of database. Algorithms like Prefix Span[2], finds frequent pattern subsequences using pseudo/projected database and BI-Directional Extension (BIDE)[3] finds frequent closed pattern mining for single data elements , none of the algorithm does closed pattern mining for multiple data sets.

In this thesis, we propose an efficient algorithm for finding the closed frequent patterns using for multiple data Item Set thus doing a general database mining, which is not limited to the number of datasets an element has. It removes the redundant pattern and gives less number efficient pattern.

`

TABLE OF CONTENTS

`

`

List of Figures

`

List of Tables

`

Chapter 1: Introduction

The tremendous growth of many business, government and scientific databases has produced a huge workspace for data analyst to interpret and digest this data. Frequent item sets play a very important role in many data mining tasks that seeks to find interesting and understandable patterns from databases, such as correlations, association rules, sequences, classifiers, episodes, clusters and many more.

It is required that, once the data is there to find ways of extracting information out of them and patterns of users' behavior must be found. The answer of this problem was the development of various data mining techniques, which is the basic subject of our thesis. Recently, data mining attracted a lot of research attention.

Data Mining is to find valid and potentially useful pattern in data which must be understandable [4].

This sequential mining problem was first introduced in [5]; two sequential patterns examples are: "80% of the people who buy a television also buy a video camera within a day", and "Every time Microsoft stock drops by 5%, then IBM stock will also drop by at least 4% within three days". The above illustrated patterns can be used to make the efficient use of shelf space for customer convenience, or to properly plan the next step during an economic crisis. Sequential pattern mining is very important for analyzing biological data [6] [7], in which range from a very small alphabet to long patterns with a typical length of few hundreds or even thousands, frequently appear.

`

Frequent pattern mining, as one of the several data mining tasks, have a big share in the data mining research. This is attributed to its wide area of applications. Applications of these frequent pattern mining, a wide area of business from market basket analysis, to analysis of promotions and catalog design and form designing store layout to customer segmentation based on their buying patterns [8].

The problem of discovering all frequent patterns can be further decomposed into two sub problems [8]:

Find all pattern of items (item sets) that have support above the minimum support, these are the frequent itemsets. Itemset other than frequent Itemset are called infrequent itemsets. We Use the frequent itemsets to further generate the patterns.

There is a wide agreement among the literature that the first sub-problem is the more important of the two. This is because it is very time consuming due to the huge search space(the power set of the set of all items) and the pattern generation phase can be done in even main memory in a straightforward manner once the frequent itemsets are found [s]. That is the reason, the great attention researchers paid to this problem in the recent years.

Although there are so many problems related to sequential pattern mining explored, we realize that the general sequential pattern mining algorithm development is the most basic one amongst all because all the others can benefit from the strategies it employs, i.e., Apriori heuristic and projection-based pattern growth. Hence we aim to develop an efficient and improved general sequential pattern mining algorithm in this paper.

Much work has been carried out on mining frequent patterns, as for example, in [1],[5],[9], [10]. However the scope of improvement is that, all of these works suffer from the problems of having a large search space and ineffectiveness in handling dense data sets, i.e., biological data. In this work, we are going to propose new strategies to reduce the

`

space necessary to be searched. Instead of searching the entire projected database for each item, as Prefixspan [2].

In this thesis, we expand the horizon of frequent pattern mining by introducing an efficient algorithm for mining the closed frequent patterns for multiple itemsets.

1.1 Motivation

Sequential pattern mining has became an important data mining task, and it has been used in a broad range of applications, which includes the analyses of customer purchase behavior, disease treatments, DNA sequences, Web access patterns and many more.

A host of technological advances have resulted in generating a huge amount of electronic data, and have enabled the data to be captured, processed, analyzed, and stored rather inexpensively. This capability has enabled many industries and innovations that generate huge volumes of electronic data such as [11]

- Banking, insurance, financial transactions - electronic banking, ATMs, credit cards, stock market data
- Supermarket check-out scanner data, point-of-sale devices, barcode readers
- Healthcare - pharmaceutical records
- Communications - telephone-call detail records
- Location data - GPS, cell phones
- Internet and e-commerce - Web logs, click-streams

An important new trend in information technologies is to identify meaningful and understandable data collected in information systems. As this knowledge is captured, this can be a key to gaining a competitive advantage over other competitors in an industry that inspires us to explore the field of frequent pattern mining.

`

One of the biggest challenges that the analysts face in today's world is to find the frequent patterns from continuously increasing data set efficiently. It has been recognized that by decreasing minimum support, the number of frequent sequential patterns can grow rapidly. This large number of frequent sequential patterns can hence reduce the efficiency, and effectiveness of the mining task. The efficiency is reduced, because of the large number of patterns generated in the first stage needed to be processed in the later stages of the mining task. Also, effectiveness is also reduced, because users have to go through a large number of elements in the result set to find useful information. In the past few years many algorithms have been made and the motivation for our work comes from the study of these algorithms in data mining for finding the frequent patterns. Each of these algorithms has contributed to different enhancements in frequent pattern mining. But they also give a huge number of frequent patterns. They are occupying more space in the memory. There are some traditional algorithms which provide the closed frequent patterns, which are redundant and few in numbers. But the problem is that they work only for single itemsets. So in this thesis we are proposing a technique by which we can get closed frequent patterns for multiple itemsets.

1.2 Research Objective

This thesis reports on our approach to frequent pattern mining. With respect to this, it explores data mining techniques that could be applied to large data sets to discover users' patterns and identify the frequent patterns itemsets. The problem statement is: **"To propose an efficient algorithm for finding the frequent closed patterns from a large incremental multiple items dataset and comparing its performance with earlier algorithms like prefix span using various real datasets".**

`

1.3 Related Work

Frequent pattern mining was introduced by Agrawal et al. (1993) for analyzing the market basket in the form of association rule mining. It analyses customer the buying habits of customer by finding associations between the different items that customers place in their "shopping baskets". For instance, if customers are buying milk, how likely are they going to also buy bread on the same trip to the supermarket? This information can help retailers to increase their sales and do selective marketing and arrange their shelf space.

The sequential item set mining problem was firstly proposed by Agrawal and Srikant, and the same was developed as a filtered algorithm, GSP [1], based on the Apriori property [12]. Since then, many sequential item set mining algorithms are being developed for Efficiency. Some are SPADE [13], Prefixspan [2], and SPAM [14]. SPADE is on principle of vertical id-list format and it uses a lattice-theoretic method to decompose the search space into many tiny spaces, on the other side Prefixspan implements a horizontal format dataset representation and mines the sequential item sets with the pattern-growth paradigm: grow a prefix item set to attain longer sequential item sets on building and scanning its database. The SPADE and the Prefixspan highly perform GSP. SPAM is a recent algorithm used for mining lengthy sequential item sets and implements a vertical bitmap representation. Its observations reveal, SPAM is better efficient in mining long item sets compared to SPADE and Prefixspan but, it still takes more space than SPADE and Prefixspan. Since the frequent closed item set mining [15], many capable frequent closed item set mining algorithms are introduced, like A-Close [15], CLOSET [20], CHARM [17]. Many such algorithms are to maintain the ready mined frequent closed item sets to attain item set closure checking. To decrease the memory usage and search space for item set closure checking, two algorithms, TFP [18], implement a compact 2-level hash indexed

`

result-tree structure to keep the readily mined frequent closed item set candidates. Some pruning methods and item set closure verifying methods, initiated that can be extended for optimizing the mining of closed sequential item sets also. CloSpan is a new algorithm used for mining frequent closed sequences [19]. It goes by the candidate maintenance-and-test method: initially create a set of closed sequence candidates stored in a hash indexed result-tree structure and do post-pruning on it. It requires some pruning techniques such as Common Prefix and Backward Sub-Item set pruning to prune the search space as CloSpan requires maintaining the set of closed sequence candidates, it consumes much memory leading to heavy search space for item set closure checking when there are more frequent closed sequences. Because of which, it does not scale well the number of frequent closed sequences. BIDE [3] is another closed pattern mining algorithm and ranked high in performance when compared to other algorithms discussed. BIDE projects the sequences after projection it prunes the patterns that are subsets of current patterns if and only if subset and superset contains same support required. But this model is opting to projection and pruning in sequential manner. This sequential approach sometimes turns to expensive when sequence length is considerably high.

1.4 Scope of the work

In this thesis, a new algorithm for frequent closed pattern mining for **multiple data sets** is proposed. This algorithm gives less number of patterns and takes less memory for execution as compare to traditional algorithm like Prefix span. This algorithm is proposed for **multiple item set.** It also supports the incremental nature of database.

Our proposed algorithm constructs a frequent sequence tree like compact tree structure only for the frequent items in a database with only one database scan. The key benefit of

`

the proposed algorithm is that it requires at most one scan of the database to find the frequent patterns. Our experimental analysis using real data sets shows that our proposed algorithm is highly efficient in terms of number of closed frequent patterns and memory than earlier algorithms.

1.5 Organization of the thesis

In this chapter, we have highlighted the problems faced by users in the frequent pattern mining and the evolution in data mining which serves as the motivation for the work reported in this thesis. Furthermore we have also outlined the specific objective of our research and related research work that has occurred in the past.

Chapter 2 provides an overview of related works in the field of data mining with brief explanations of data mining techniques. It also briefly introduces the Sequential pattern mining with its problem definition and benefits. Finally, we present studies that employ the algorithms to find the frequent itemsets which help to position our work in its context in the following chapters.

Chapter 3 introduces our approach to frequent itemsets mining and discusses some of the tasks carried out with respect to the objective outlined. It also includes the limitation of earlier approach towards frequent pattern mining.

Chapter 4 presents the performance study conducted on the proposed algorithm. Each conducted experiment is discussed and detailed comments on the results are given.

Finally, Chapter 5 concludes the thesis and gives some suggestions for future work.

`

Chapter 2: Literature Review

2.1 Basic Concepts of Data Mining:-

2.1.1 What is Data Mining?

We can simply be explained that Data Mining (DM) is a self automated process for discovering uncovered knowledge from huge amount of data. It involves complex Data Structures, Algorithms, Statistics and Artificial Intelligence [20].

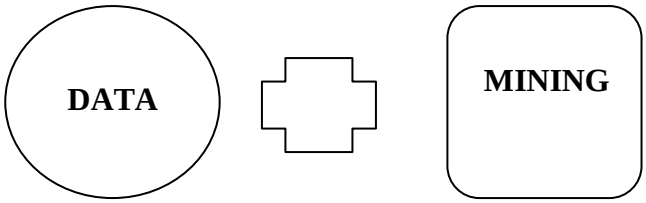

Fig. 2.1: Mine or extract the data for important knowledge

Data mining [Chen et al. 1996] is the process of extracting interesting & previously unknown and useful information from large information repositories like relational database, data warehouses etc. Also data mining is known as one of the core processes of **Knowledge Discovery in Database** (KDD).

Usually there is a sequence of processes:-

1.Data cleaning:- A process by which remove unnecessary data.

2.Data integration:- A process to integrate the multiple data sources.

3.Data selection:- A process to retrieve the data from the databases.

4.Data transformation:- A process by which data transformed or consolidated into forms which are appropriate for mining by performing summary or aggregation operations,for instance.

`

5.Data Mining:- A process in which efficient methods are applied in order to extract importent data patterns.

6.Pattern Evaluation:- A process to identify the truly itresting patterns which are representing knowledge.

7.Knowledge presentation:- A process where visualization and knowledge techniques are used to represent the mined knowledge to the user.

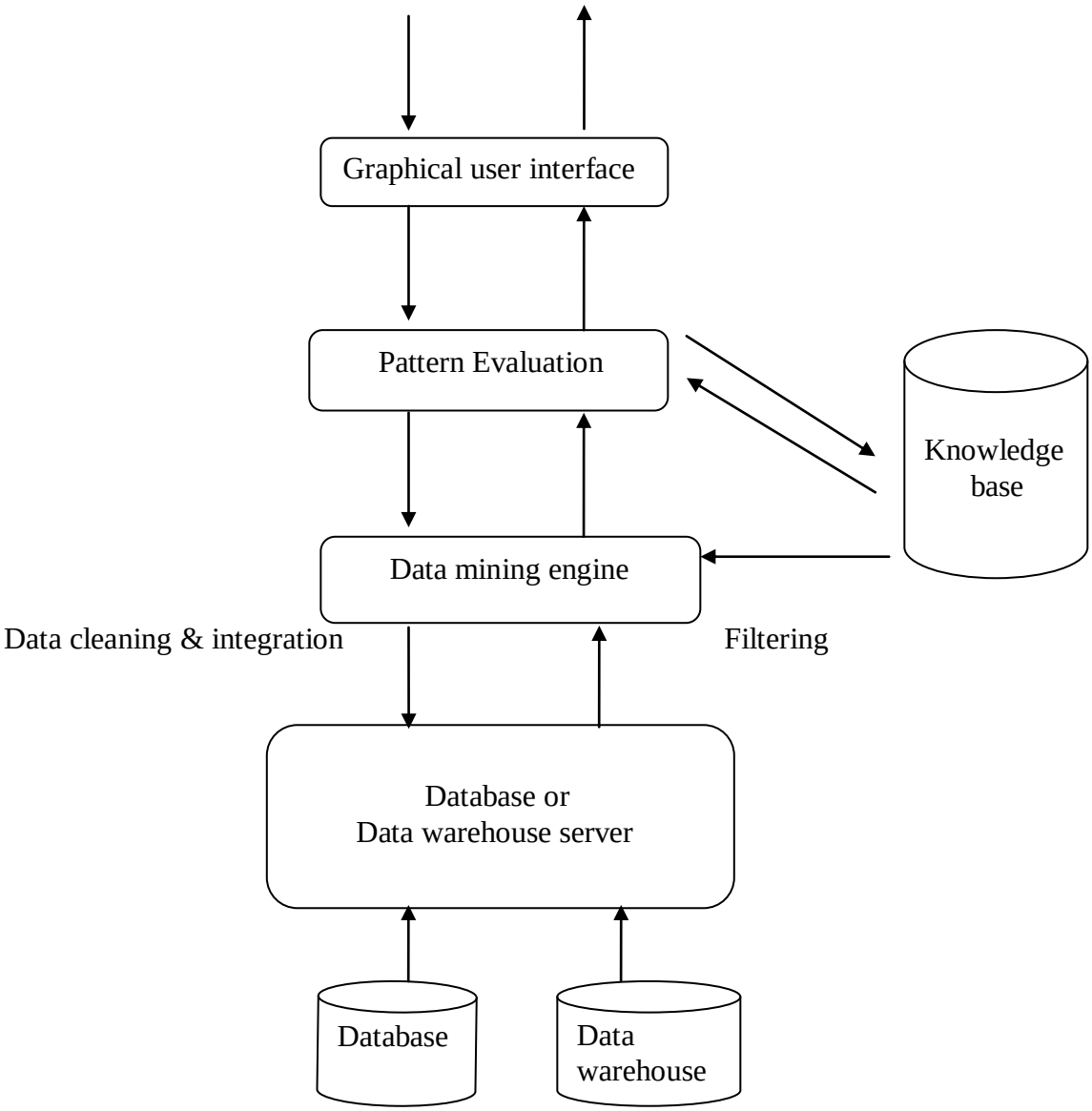

Fig. 2.2: Architecture of Data mining system [37]

`

Database & data warehouse:-they are the one or a set of the databases, data warehouses or other kind of data repositories in which we store the data in which we apply mining to get some useful information.

Database or data warehouse server:-The work of this server is extracting the relevant data based on the user's request.

Knowledge base:-It includes hierarchies, used to organize attributes or attributes values into different levels of abstraction. E.g. Metadata, which describes data from multiple heterogeneous sources.

Data mining engine:-It contains a set of functional modules for tasks like characterization, association, classification, cluster analysis, and evolution and deviation analysis.

Pattern evaluation module:-This step works with interesting constraints and interacts with the data mining modules to focus the search towards interesting patterns. It may use to decide the thresholds to filter out discovered patterns.

Graphical user interface:-This module is used for better communication between users and data mining system. It allows the user to interact with the system by specifying a data mining query.

Knowledge Discovery in Database also known as KDD a synonym of Data Mining, which comprises of three stages [20]:

`

- The understanding of business and data.
- Performing the pre-processes tasks.
- Data Mining and Reporting

.2.1.2 Data Mining Background, Research and Evolution

In today's world, the improved processing power and sophisticated technologies has increased the business need, and now people expect a lot from systems [20]. These days, the computer systems are not just used for storing data only but also for providing forecasting and information. Data Mining is the root of a concept, which has been recently, introduced known as BI or Business Intelligence [20]. The need is to extract knowledge out of the abstract data. With recent technical advances in processing power, memory, interconnectivity data mining is seen as an increasingly important tool by modern business to transform abstract data into business intelligence form giving an additional advantage [20].

The rise of data mining originated from the emergence of data warehouse [21]. As early as 1990s, in "Building the Data Warehouse", William H. Inmon--the U.S. information engineering [21] professional introduced the concept and implementation steps of data warehouse. The concept is that. Data Warehouse is a subject oriented, nonvolatile, integrated, and time variant collection of data, in support of management's decision-making [21]. In his monograph, he classified the implementation steps into three major parts: data preprocessing, data mining and KDD.

`

After 20 years of development, while confirming the accuracy of the definition, a number of applications have further confirmed the importance of "Building the Data Warehouse" Data processing is the core to establish data warehouse; Data mining is considered as the important technology and method [21]; KDD is regarded as the essential process and method of data warehouse and data mining is one important step of KDD. From the above, we can see that the major technology and core to establish a complete data warehouse is whether the data mining model system is in common use [21].

Data mining application are characterized by the ability to deal with the expansion of business data and accelerated market changes, these characteristics help providing effective tools for decision makers, such tools can be used by business users (not only statisticians) for analyzing huge amount of data for patterns and trends [22]. Consequently, data mining has become a research area with increasing importance and it involved in determining useful patterns from collected data or determining a model that fits best on the collected data [22]. Different classification schemes can be used to categorize data mining methods and systems based on the kinds of databases to be studied, the kinds of knowledge to be discovered, and the kinds of techniques to be utilized [22].

Data mining techniques used in business-oriented applications are known as Business intelligence (BI) [22]. BI is a general term to explain all processes, techniques, and tools that gather and analyze data for the purpose of helping enterprise users to make better decisions [22]. The difficulty of discovering and deploying new knowledge in the BI context is due to the lack of efficient and complete data mining system [22]. The measure of any business intelligence solution is its ability to derive knowledge from data. The

`

challenge is fulfilled with the ability to identify patterns, trends, rules, and relationships from volumes of information which is too large to be processed by human alone.

2.1.3 Introduction to Data Mining Techniques

The most important data mining techniques are as follows:

2.1.3.1 Clustering

Clustering[] is another method for categorizing objects based upon their attributes.

Clustering approaches attempt to group items together that are very similar to members of their own cluster, but vastly different from items in other clusters.

Clustering is a process of partitioning a set of data (or objects) in a set of meaningful sub-classes, called cluster. Cluster is a collection of data objects that are similar to one another and thus can be treated collectively as one group but as a collection, they are sufficiently different from other groups. Clustering is a unsupervised classification which means we do not know the class labels and may not know the number of classes. The main difference between clustering and classification is that there are no preset rules about what constitutes a group of items. Clustering has wide applications in Pattern Recognition, Spatial Data Analysis, Image Processing, Market Research, Information Retrieval, Web Mining etc.

2.1.3.2 Classification

Classification is a process that seeks to group together objects with similar attributes. When classifying a new object, its attributes are compared with the attributes of objects in existing groups, and the object is assigned to the group whose attributes are most similar.

Classification is the task of mapping a data item into one of several predefined classes. In the Web domain, one is interested in developing a profile of users belonging to a particular

`

class or category. This requires extraction and selection of features that best describe the properties of a given class or category. Classification can be done by using supervised inductive learning algorithms such as decision tree classifiers, naive Bayesian classifiers, k-nearest neighbor classifiers, Support Vector Machines etc. For example, classification on server logs may lead to the discovery of interesting rules such as: 30% of users who placed an online order in /Product/Music are in the 18-25 age groups and live on the West Coast.

2.1.3.3 Association Rule

Association rule discovery techniques are generally applied to databases of transactions where each transaction consists of a set of items. In such a framework the problem is to find all associations and correlations among data items where the occurrence of one set of items in a transaction implies (with a certain degree of confidence) the presence of other items. One of the ways in which this goal may be realized is by understanding what products or services customers tend to be purchased at the same time, or later as follow-up purchases. As such, determining consumer purchasing behavioral trends is a very common application of data mining, and association and sequencing techniques can perform this kind of analysis. For example, using association rule discovery techniques we can find correlations such as the following:

- 40% of clients who accessed the Web page with URL /company/product1, also accessed /company/product2; or
- 30% of clients who accessed /company/special placed an online order in /company/product1.

`

2.1.3.4 Sequential Patterns

Sequential patterns are another form of a frequently used application in data mining. Sequential pattern mining functions are quite powerful and can be used to determine the set of customers associated with some frequent buying pattern. A sequential pattern function tries to detect frequently occurring patterns in the data. As an example, if customers increase their purchase of a particular product X by 20% and also increase their consumption of another product Y by 25%, then product Z is found to also increase by 10%. Whether product's Z increase in purchases was a direct result of the increase in sales of products X & Y collectively is uncertain; what is certain is that this relationship holds and it is up to the data miner to examine this trend further by issuing further queries to determine whether or not a particular relationship is a casual one. In this manner, sequential patterns are used in determining frequency of events occurring and their relationship to other events.

2.2 Sequential Pattern Mining

2.2.1 Problem Definition

The problem of finding can be stated as follows [2]: Given a sequence database and the Min_support threshold, sequential pattern mining is to find the complete set of sequential patterns in the database.

An example of sequential pattern mining is: Let sequence database be S given in Table 2.1 and min_support = 2. The set of items in the database is { 1, 2, 3, 4, 5, 6, 7 }.

`

Sequence_ID	Sequence
1	<1(1 2 3)(1 3)4(3 6)>
2	<(1 4)3(2 3)(1 5)>
3	<(5 6)(1 2)(4 6)3 2>
4	<5 7(1 6)3 2 3>

Table 2.1: A Sequence Database

A sequence <1(1 2 3)(1 3)4(3 6)> has five elements: (1), (1 2 3), (1 3), (4), and (3 6), where items 1 and 3 appear more than once, respectively, in different elements. It is a nine-sequence since there are nine instances appearing in that sequence. Item 1 happens three times in this sequence, so it contributes 3 to the length of the sequence. However, the whole sequence <1(1 2 3)(1 3)4(3 6)> contributes only one to the support of 1. Also, sequence <1(2 3)4 6> is a subsequence of <1(1 2 3)(1 3)4(3 6)>. Since both sequences 1 and 3 contain subsequence s= <(1 2)3>, s is a sequential pattern of length 3.

2.2.2 Benefits and applications

Since we have a database which contains a large number of sequences, and users have different interests and requirements, to find the most interesting and understandable sequential patterns, usually a minimum support is pre-defined by users. By using the minimum support we can prune out those sequential patterns which have no interest and make the mining process more efficient[29]. Obviously a higher support of sequential

`

pattern is desired for more useful and interesting sequential patterns. However some sequential patterns that do not satisfy the support threshold are still interesting.

Applications: Most data and applications are time-related

- Customer shopping sequences[30]:

  E.g., first buy computer, then CD-ROMS, software, within 3 mos.

- Telephone calling patterns
- Natural disasters (e.g., earthquake, hurricane)[32]
- Disease and treatment[33]
- Stock market fluctuation[31]
- Weblog click stream analysis[34]
- Healthcare Data Mining Applications[35]

2.3 Algorithms for finding the frequent patterns

2.3.1 Apriori Algorithm [23]

The Apriori algorithm was appeared first in 1994[23] and remain the standard reference of all algorithms for finding association rules. It outperformed the previous algorithms by a great margin. The major difference in Apriori was the much less candidate set of itemsets it generates for testing in every database pass. Here comes the time saving. The algorithm benefited of the fact that for an itemset to be frequent, all its subsets must be frequent.

`

The algorithm starts by collecting all the frequent 1-itemsets in the first pass. It uses this set (called L_1) to generate the candidate sets to be frequent in the next pass (called C_2) by joining L_1 with itself. The trick here is that the algorithm eliminates from C_2 any set that has a subset not in L_1 , because it knows a-priori that it can't be frequent, hence reducing its size dramatically. The algorithm proceeds in the same manner generating the candidates of size k from the large itemsets of size k-1, then reduces the candidate set by eliminating all those which have infrequent k-1 subsets, then counts occurrences of the remaining candidates in the next pass to find the frequent k itemsets. The algorithm terminates when there are no candidates to be counted in the next pass.

The key concepts in this algorithm are:

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

Pseudo-code:

C_K: Candidate itemset of size k

L_K: frequent itemset of size k

L_1= {frequent items};

For (k=1; L_k != ∅; k++) **do begin**

C_{K+1}= candidates generated from L_K;

for each transaction t in database do

`

Increment the count of all candidates in C_{K+1} that are contained in t

L_{K+1} = candidates in C_{K+1} with min support

End

return $U_k L_k$

Advantages of Apriori algorithm [25]:

- Uses large item set property.
- Easily parallelized
- Easy to implement

Limitations of Apriori algorithm [25]:

Apriori algorithm, in spite of being simple, has some limitation. They are:

- It is costly to handle a huge number of candidate sets.
- 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, an efficient FP-tree based mining method, FP-growth, which contains two phases, where the first phase constructs an FP tree, and the second phase recursively Researches the FP tree and outputs all frequent patterns**.**

2.3.2 Prefix span[2]

PrefixSpan [2] is a recently proposed method for mining frequent sequential patterns.

Unlike Apriori-based algorithms, such as GSP [1], that mine frequent sequential patterns by candidate generation, PrefixSpan uses prefix projection to mine the complete set of frequent sequential patterns. Here is a brief description on how does PrefixSpan work. For more detailed information, I refer the reader to [4].

Given sequence database s, and minimum support of x , PrefixSpan performs the

`

following steps to mine the frequent sequential patterns.

Step 1: Scan s once, and find all the frequent items. These frequent items are frequent sequential patterns with length one.

Step 2: According to the length-1 frequent sequences found in the first step, the complete set of frequent sequential patterns can be divided into different subsets, one subset for each length-1 frequent sequence.

Step 3: Each subset, in second step, can be mined by constructing its corresponding postfix projected database, and mining it recursively (a frequent item's, e, projected database is the set of postfixes of e in the original database).

PrefixSpan uses recursion to mine the frequent sequences. First frequent items in the database are found, and then for each frequent item a projected database is created (the prefix of the projected database is a frequent sequence). This process is repeated for each projected database, until the projected database contains no frequent items, which in that time the recursion for that branch ends, and the execution returns to the calling procedure. Let's call this end of the recursion branch as a backtrack. Algorithm is from [4], and shows how does Prefixspan work.

Input: A sequence database s, and minimum support threshold x.

Output: The complete set of sequential patterns.

Method: Call PrefixSpan(<>, 0, s)

Subroutine PrefixSpan(α, l, s |α)

Parameters: α: a sequence; l: the length of α; s|a: the a-projected database, if α ¹ <>, otherwise the sequence database s .

`

Method:

Scan s |α once, find the set of frequent items ƀ such that:

ƀ can be assembled to the last element of a to form a sequence;

or < ƀ > can be appended to a to form a sequence.

For each frequent item b, append it to a to form a sequence α', and output α';

For each α', construct α'-projected database s |α', and call PrefixSpan(α', l + 1, s |α').

In the following chapters I describe two programs for mining frequent Max, and Closed sequential patterns. These programs use PrefixSpan as their basis to generate candidate frequent Max, and Closed sequences.

2.3.3 BI-Directional Extension **(BIDE) [3]**

An efficient algorithm for discovering the complete set of frequent closed sequences.

- BI-Directional Extension: new paradigm for mining frequent closed sequences without candidate maintenance.
- Back-Scan pruning method.
- Back-Skip optimization technique.
- Performance study: BIDE can be over an order of magnitude faster than the previous algorithms and consumes orders of magnitude less memory.

Steps of Bide Algorithm:

- Scan the database once in order to find all the frequent 1-sequences.
- For each frequent 1-sequence build a pseudo-projected database and check if it can be pruned: Back-Scan.

`

- For every sequence that cannot be pruned compute the number of backward-extension items and then call subroutine *bide*.
- Subroutine bide:
    - For prefix S_p scan its projected DB.
    - Compute the number of forward-extension items.
    - If there is no forward-extension neither item nor backward-extension item, S_p is closed.
    - Grow S_p with each locally frequent item to get a new prefix.
    - Build the pseudo-projected DB for the new prefix.
    - For each new prefix:
    - Check if it can be pruned

If not, compute the number of backward-extension items and call itself

Benefits of BIDE:

- BIDE can significantly outperform Prefixspan and SPADE.
- BIDE consumes much less memory and can be faster than CloSpan.
- BackScan pruning method is effective in enhancing the performance.
- Prunes search space more deeply

Limitations of BIDE:

As we know that BIDE is much efficient algorithm for mining the closed frequent pattern very efficiently. But there is a limitation in BIDE that it can mine dataset which contains the single itemsets.

`

2.4 Summary

In this chapter, we have surveyed the broad area of Data Mining with a specific focus on Sequential pattern mining. We have also introduced various techniques of data mining. The problem definition of Sequential pattern mining and its benefits and application are explored in detail. We have also surveyed many of the earlier algorithms for frequent pattern mining like Apriori, prefix span, BIDE etc. and also explained Apriori, prefix span, BIDE in detail using pseudo-code and example. We found that all the earlier algorithms are either lack in memory or having large number of patterns. So we have proposed a new efficient algorithm in the next chapter.

`

Chapter 3: Proposed Algorithm for Frequent closed Pattern Mining

The objective of this chapter is to introduce our approach to close frequent pattern mining and to provide details of the work carried out in acquiring closed frequent patterns from the multidata itemset. In particular, we will suggest our approach to give a time efficient algorithm for finding the frequent patterns from a large data set.

The chapter is organized as follows. Section 3.1 gives the detailed description of the traditional algorithm. In Section 3.2 we are discussing about the proposed algorithm which can meet with our goal. Section 3.3 provides the comparison of the proposed algorithm with the earlier algorithms. Finally section 3.4 summarizes the chapter.

3.1 Traditional Approach

In traditional approach algorithm , a frequent sequence tree is constructed for itemsets based on lexo-graphical ordering , these algorithm generate prefix sequences and list all the prefix subsequence possible and calculate their support , like prefix span algorithm , hence consumes a lot of memory and time. Other traditional algorithms try to predict the closed frequent sequences from the current sequences and decide whether to explore the sequence further or not but these algorithms are defined for itemset having only single data and hence have less use in real scenarios.

Here we try to propose a closed frequent set algorithm for itemset having multiple data, thus overcoming limitation of traditional algorithms.

`

3.2 Proposed Algorithm

Taking an inspiration from traditional algorithms, we here propose an algorithm for multidata itemsets /sequences.

3.2.1 Basics of algorithm

Frequent sequences / itemset are sequences for which the support in database is above threshold level [2].

Closed Frequent sequences/itemsets, sequences for which no such super sequence exist which has support equal to this sequence. To find closed frequent sequences, we need only explore/grow sequences using its locally frequent items [3].

Support of a pattern= no. of occurrence/ size of database

3.2.1.1 Frequent sequence tree

Frequent sequence tree [3] is constructed to list all the frequent sequences/itemsets. It enlists all the prefix sequences in lexographical ordering, for example if there exits elements A, B, C in lexographical order. Frequent sequence tree list all the possible combination of prefix sequences.

`

A simple frequent sequence tree will look like:

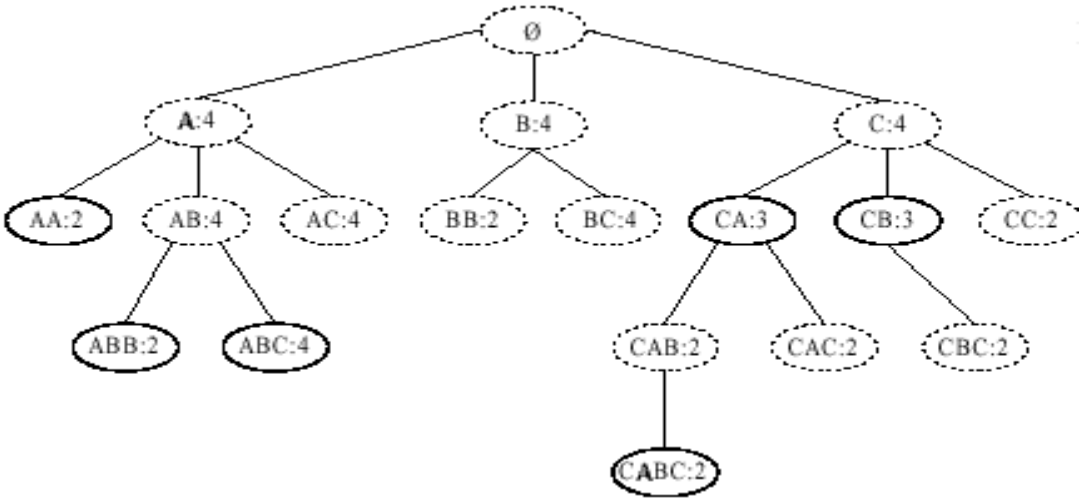

Fig.3.1: The lexicographic frequent sequence tree[3]

To list all the frequent sequences/itemsets all the possible combinations need to be explored and its support need to be calculated thus is a memory consuming task. Moreover, this approach gives the redundant Sequences as output. To reduce the memory and Redundancy Closed frequent sequences need to be Detected which stops further exploration of Sequences if they are found to be closed thus reducing memory required and eliminating redundant solutions.

Initially we would explain an efficient algorithm for generation of closed frequent Sequences having single data in itemsets taking inspiration from traditional Algorithm which will be used in proposing our algorithm.

`

3.2.2 Approach for generations of closed sequences having single item in datasets/element.

For Computation of locally frequent items a projected database is constructed for every sequence, projected database comprises of projected sequences. These sequences are calculated by subtracting first instance of prefix sequence from complete sequence, two kinds of projection techniques are used:-

1) physical projection
2) pseudo projection

In physical projection, the projected database is physically constructed and stored in memory.

In pseudo projection [2], only pointers to projected sequence are stored.

After generating initial prefix sequences, for every prefix sequence, it is required to estimate whether the sequence is closed or not to further stop growing the sequence, its closed if it has no super sequence that has support equal to this sequence.

A super sequence of current sequence can be generated by adding element in front, middle or back of sequence.

Thus, if no element can be added to the sequence which generates support equal to support of current sequence, the sequence is said to be closed and no further growth is required.

Two techniques can be used to check whether the sequence is closed or not:-

1) Forward Enumeration
2) Backward Enumeration

In Forward enumeration, locally frequent itemset/elements with respect to prefix sequence/itemset are explored/ enumerated and support is calculated from projected database.

`

In Backward enumeration, elements are inserted before and between the sequences to calculate support for new sequence.

Forward enumeration technique:-

In forward enumeration, complete set of locally frequent items are explored to check if any item exist whose support is equal to support of prefix sequence.

For example, if database given where number 1, 2, 3 are represented as items in a sequence

Sequence_ID	Sequence
1	3 1 1 2 3
2	1 2 3 2
3	3 1 2 3
4	1 2 2 3 1

Table 3.1. An example of sequence database

For sequence 1 2

First instances are {3 1 1 2, 1 2, 3 1 2, 1 2}

Projected sequence {2 3 1} and projected database-{3, 3 2, 3, 2 3 1}

Thus it can be seen that 3 has support equal to support of given prefix sequence 1 2, hence 1 2 is not closed and required to be further grown. This way forward enumeration can be used to find whether sequence is closed or not.

`

Backward enumeration technique-

For backward enumeration, items can be inserted at two places, at the start of sequence or the middle of sequence.

To check at all positions in sequence, which can contain items in between and before prefix sequence?

Last occurrence of prefix sequence, in sequence is to be found and known as its last instance.

For example in sequence s=1 2 2 3 1, last instance of s_p =1 2 is given by 1 2 2.

From this place, all the position between first and last instance of prefix sequence (s_p) in sequence(s) is to be checked to find the occurrence of other elements/items in between and calculate support for each.

To find items in between ,the positions between the end of first instances of previous prefix and i^{th} last appearance of prefix sequence item is checked and is known as i^{th} maximum period of a prefix.

For example in sequence 1 2 3 2, and prefix sequence 1 2

The first instance of 1 is 1.

The first instance of 2 is 12.

And last appearance of 1, at position number 1

And last appearance of 2, at position number 4

Thus, 1^{st} maximum is ϕ.

2^{nd} maximum is 2 3.

`

This way all the position in between can be checked using i^{th} maximum generation of the current prefix sequence and support can be calculated .If any item exists in all the i^{th} maximum periods of prefix sequence, thus the sequence is not closed and need to be grown, and the item is said to be backward enumeration of sequence.

Thus if any item/element occurs in forward and backward enumeration, the sequence is not closed, and further enumeration /expansion is required.

To further optimize and improve the mining efficiency, we can use two techniques -

1. Back scanning and pruning
2. Scan skipping technique

Back scanning and pruning-

For back scanning and pruning, we have to find an item which exists in each of the ith semi-maximum periods of prefix sequence; if it does then we can stop growing prefix sequence.

To find i^{th} maximum periods, i^{th} last in first appearance is to be found for a prefix sequence.

Last in first appearance: of an item is the last appearance of it in first instances of prefix.

For ex. Sequence -3 1 1 2 3

Prefix sequence 3 1

First instance -3 1

2^{nd} last in first appearance is first 1 in sequence.

And 1^{st} last in first appearance is first 3 in sequence.

`

i^{th} semi maximum period:

It is the piece of sequence between the end of the first instance of prefix sequence and i^{th} last in first appearance

E.g. s=1 2 3 2

Sp =1 3,

The second maximum period of prefix 13 in s is 2 while 1^{st} semi –maximum is ϕ.

Thus, if there exist an element that exists in the all i^{th} semi –maximum periods of sp, we can stop growing prefix sequence.

Scan Skipping optimization-

In backward scanning a number of semi-maximum periods with respect to a prefix sequence is to be scanned ,to reduce the time required and optimize it ,we can follow a technique called scan skipping technique which uses a simple approach of taking intersection and determining whether to move forward or not.

In it, for a prefix sequence, there are k number i^{th} semi maximum periods, determined as $\{smp_1^i, smp_2^i \ldots . smp^i_{supsp}\}$.since an element/item should be present in all semi – maximum period, taking intersection of smp's will give unique items, thus set of items which appear in each of i^{th} semi maximum periods equals to

$\{smp_1 \cap ismp_2^i \ldots\ldots smp^i_{supsp}\}$.

We need to check if this list is empty or not, which determines that back scanning is required or not.

`

After taking intersections of first few smp's if the list becomes ϕ, thus we know ϕ ∩ anything is ϕ, and hence we don't need to check further, thus speeding up the process and providing result.

e.g. from table 3.2:

Sp=1 2 3 support =4

3rd maximum periods = {ϕ, ϕ, ϕ, 2}

After taking intersection, we know, there will be no item in 3rd maximum period, thus subsequence can be checked for 2nd and 1st maximum periods.

Algorithm-

CFP_single_data_itemsets()

1. Find Frequent items of size 1.

2. For each frequent item create pseudo projected database (it holds pointer to the location of next characters in database. E.g. ABC, ADF, AER the for item A, the pseudo projected database will hold pointers to BC, DF and ER).

3. for each frequent item check if BACKSCAN PRUNING is possible - if not possible then

    Perform backward enumeration check to get backward enumeration items (BEI).

    call CFP_single_data_itemsets_forward()

CFP_single_data_itemsets_forward()

    Find locally frequent items from Pseudoprojected database. (LFI)

    Find forward enumeration items (FEI)

    if (BEI + FEI == 0) // no extension possible, this is closed item

`

Add the frequent item to the closed items list

For each locally frequent item LFI

Create pseudo projected database

Check if Backscan pruning is possible, if not then

perform backward enumeration check

call CFP_single_data_itemsets_forward()

---------- END ----------

3.2.3. Approach for finding closed frequent pattern for multiple data item sets-

Previous traditional algorithms, can only mine closed frequent sequences of single items. In this thesis we propose an algorithm for mining closed sequences of itemsets containing multiple item.

Since this approach is an extension of traditional approach thus the basic approach like forward and backward enumeration and backscanning will remain same with modification to adapt to multidata itemsets.

For a frequent sequence to be closed in multidata itemset sequences, there could be two expansion related to prefix sequence-

1. IS- expansion when a new itemset is added to the sequence.

2. I-expansion when an item is added to any one of the itemset already in sequence.

Thus there is a conditionfor a prefix sequence to be closed there must be no forward IS or I expansion and no backward IS or I expansion possible multi data itemsets otherwise sequence is not closed.

`

1. Forward enumeration-

a) Using IS – expansion

When only new itemset, i, can be to sequence $s=i_1,i_2,\ldots..i_m$ where i^i contains or is a subset of items ($x_{i1}\ldots x_{in}$) to form a new prefix

$s=i_1, i_2, i_3, i_m, i'$ which has support equal to the support of s.

b) Using I expansion

When any new item y can be added to prefix $s=i_1,i_2,\ldots.(i_m^y)$ and which has support equal to support of s.

forward extension follows the same approach ,by searching locally frequent sequence/itemsets, of IS and I expansion to determine the prefix sequence is closed or not, if there is an item or itemset which has support equal to support of s then prefix sequence is not closed.

Finding IS and I expansion items

Example:

Sequence_ID	**Sequence**
1	(2 4) (2 3 4) (1)
2	(2) (1 2 5) (2)
3	(1 2) (2 3 4)

Table 3.2. An example sequence multiple item database

`

If sequence sp=1,there exist an item 2 which is both forward IS expansion and I expansion. Similarly for sp=12 there is no extension.

2) Backward enumeration-

A) Using IS-expansion

any itemset I that can be used to extend s to get new prefix $i_1, i_2, i_3, \ldots i_{l-1}\ x_{ie} \ldots i_m$.

(¥ $1 < l < m$) and has support equal to support of s 0.

B) Using I-expansion

When any new item y can be added to prefix s to get new prefix= $i_1, i_2 \ldots (y_{l1} .. y_{lj-1}\ y\ y_{lj} \ldots y_{lk}) \ldots i_m$ (¥ j ,$1 \leq j \leq (k+1)$)

if $1 \leq l \leq m$ where $1 \leq j < k$ if $l = m$

and has support equal to support of s.

For IS expansion ,same procedure of finding i^{th} maximum period and finding new itemsets which have support equal to support of prefix sequence,& lie in between the sequence s can be determined, since itemsets can be treated as one entity ,hence finding IS expansion is similar as in single data itemset search.

For I expansion, two definitions need to be given, to understand I-expansion, difference between two itemsets (projection of itemset i2 with respect to i2).

$I_1 = (y_{11} \ldots\ldots\ldots y_{1m})$

$I_2 = (y_{21} \ldots\ldots\ldots y_{2n})$

`

If $i_1 \leq i_2$ then $(i_2 - i_1)$ is given by all those item which are in i_2 and nor i_1 & i_2 is appearance of i_1.

i_{th} I-expansion period of a prefix sequence between the end of the first instance of prefix sequence and being of first event after the ith last in last appearance.

i_{th} backward I-expansion

Itemset of a prefix sequence is the union of projected events of event i_i with respect to any even i_j in I-expansion period using these definitions ,we can backward enumerate the sequence and check for :

If there exist 'y' an item in each of the ith I-expansion itemset of prefix sequence, thus y is an backward I – expansion.

Example: to find the backward enumeration-

In the table 3.2

Prefix sequence =2, support =2

I-expansion itemset with support of three sequences is-

$\{1,4\} \cup \{3,4\} = \{1,3,4\}$

$\{1,5\},\{1\} \cup \{3,4\} = \{1,3,4\}$

2 is the last event of prefix sequence and 3,4,5 are greater than 2 thus must be excluded from I-expansion itemsets.

`

The first I-expansion itemsets of prefix with respect to three sequence ,1,1,and 1.thus 1 is backward I-expansion with respect to prefix 2.

Using back space pruning and scan skip, the procedure can be optimized[s].

Algorithm:

CFP_multiple_data_itemsets()

1. Find frequent items of size 1.
2. For each frequent item, create pseudo projected database (it holds pointer to the location of next characters in database . e.g. ABC, ADF, AER the for item A, the pseudo projected database will hold pointers to BC, DF and ER).

3. For each frequent item check if BACKSCAN PRUNING is possible - if not possible then

   perform backward enumeration check to get backward enumeration items (BEI).

   perform backward enumeration check to get backward enumeration items for IS expansion (BEI1).

   perform backward enumeration check to get backward enumeration items for I expansion (BEI2).

call CFP_multiple_data_itemsets_forward()

`

CFP_multiple_data_itemsets_forward()

Find locally frequent items from Pseudoprojected database. (LFI)

Find forward enumeration items (FEI)

perform forward enumeration check to get forward enumeration items for IS expansion (FEI1).

perform forward enumeration check to get forward enumeration items for I expansion (FEI2).

if (BEI1 + BEI1 + FEI1 + FEI2 == 0) // no extension possible, this is closed item

Add the frequent item to the closed items list

For each locally frequent item LFI // repeat same steps as done for the Frequent items of size 1

Create pseudo projected database

check if Backscan pruning is possible, if not then

perform backward enumeration check

call CFP_multiple_data_itemsets_forward()

---------- END ----------

`

3.3 Summary

In this chapter, we have identified the drawbacks in the traditional methods used to solve the frequent pattern mining. With respect to this, we have introduced our approach to frequent pattern mining by proposing an efficient algorithm and explain it in detail with example and algorithm. We also described the advantage of our approach and finally chapter ends with the comparison of the proposed algorithm with the existing methods.

`

Chapter 4: Implementation and Experimental Results

4.1 Environmental Setup

We have used the following configuration while finding the experimental results

4.1.1 Hardware Configuration

Processor	:	Intel Core 2 Duo
Processor Speed	:	2.20GHz
Main Storage	:	3GB RAM
Hard Disk Capacity	:	250GB
Monitor	:	Samsung 17”5’ Color

4.1.2 Software Configuration

Operating System	:	Windows 7
Front end	:	Java
Back end	:	Datasets (explained in 4.2)

4.2 Datasets

In our proposed system, we had assumed real datasets which are used to find the experimental results of the comparison of our proposed algorithm with the earlier algorithms. This real dataset is further elaborated as given below in the next section.

4.2.1 Real Datasets

We have used real datasets namely BMS-Web View1 [26] to prove the efficiency of our proposed algorithm.

4.2.1.1 BMS-WebView-1

The BMS-WebView-1 datasets contains several months’ worth of clickstream[36] data from two e-commerce web sites [27]. Each transaction in this dataset is a web session

`

consisting of all the product detail pages viewed in that session. That is, each product detail view is an item. The goal of dataset is to find associations between products viewed by visitors in a single visit to the web site. We are making the BMSWebView-1 dataset available to the research community. This dataset comes from a small dot-com company called Gazelle.com, a legwear and legcare retailer, which no longer exists; a portion of their data was used in the KDD-Cup 2000 competition [28]. The access information is available from the KDD-Cup 2000 home page at http://www.ecn.purdue.edu/KDDCUP.

4.3 Analysis and Results

In this section, we present the experimental analysis and results on the performance of our proposed pattern mining algorithm. We compare the performance of our proposed pattern mining algorithm with that of Prefix Span in terms of number of resultant pattern, memory requirement and time consumption.

We obtain consistent results for all of the above datasets. We divide the experimental analysis in three parts: in the first part we examine the number of resultant pattern ,in the second part we examine the performance in memory requirement and in the third part we study the performance comparison in runtime.

4.3.1 Resultant Patterns Comparison

We study the number of resultant patterns of the proposed pattern mining algorithm with the variation of support threshold on our dataset as described above. The results are presented in Fig. 4.1(table 4.1), which indicates the number of resultant patterns. The *x*-axis in the graph in the Fig 4.1 depicts the change in support threshold in each case. It can

`

be observed from the figures that, the lower the minimum support value, the higher the number of patterns. The reason is that, with lowering the minimum support value, more and more itemsets become frequent, and the number of patterns increases accordingly.

Support (%)	Proposed Algorithm	Prefix span
.1	178	670
.2	69	89
.3	14	15
.4	7	7
.5	0	0

Table 4.1: Number of Pattern Comparison on *BMS-WebView-1* data set

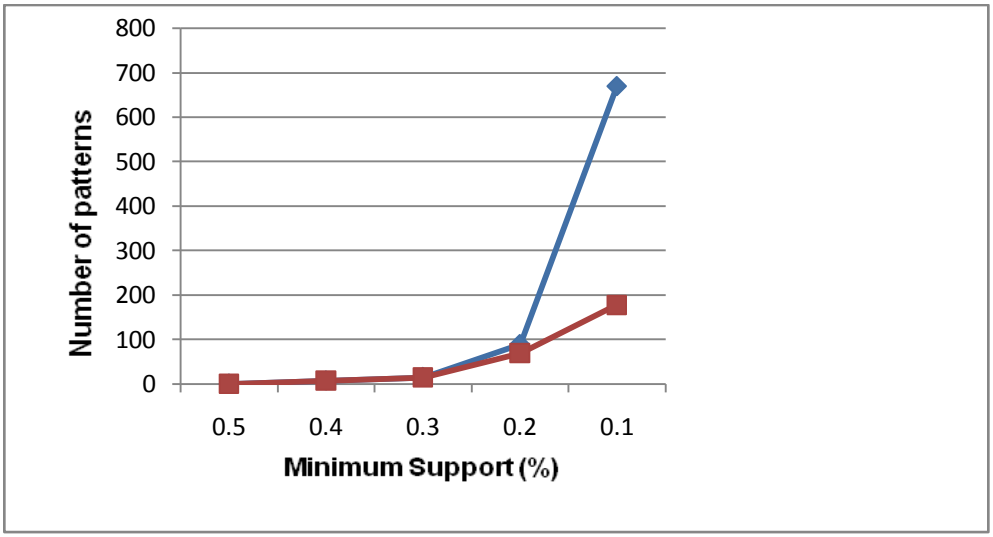

Fig: 4.1: Number of pattern Comparison on *BMS-WebView-1* data set

`

4.3.2 Memory Comparison

We study the memory requirement of the proposed pattern mining algorithm with the variation of support threshold on our dataset as described above. The results are presented in Fig. 4.2(also table 4.2), which indicates the size of maximum memory used. The *x*-axis in the graphs in the Fig 4.2 depicts the change in support threshold in each case. It can be observed from the figures that, the lower the minimum support value, the higher the memory requirements. The reason is that, with lowering the minimum support value, more and more itemsets become frequent, and the number of patterns increases accordingly. However, for the graph in Fig. 4.2 we can observe that proposed pattern mining algorithm requires reasonably less amount of memory to store the patterns in all datasets presented in the experiment than memory requirement in the Prefix Span. the items are stored in lexicographical order than frequency descending order.

Support (%)	Proposed Algorithm	Prefix Span
.1	31.64	46.63
.2	22.58	19.26
.3	16.47	17.78.
.4	12.10	9.92
.5	0.0	7.71

Table 4.2: Memory Comparison on *BMS-WebView-1* data set

`

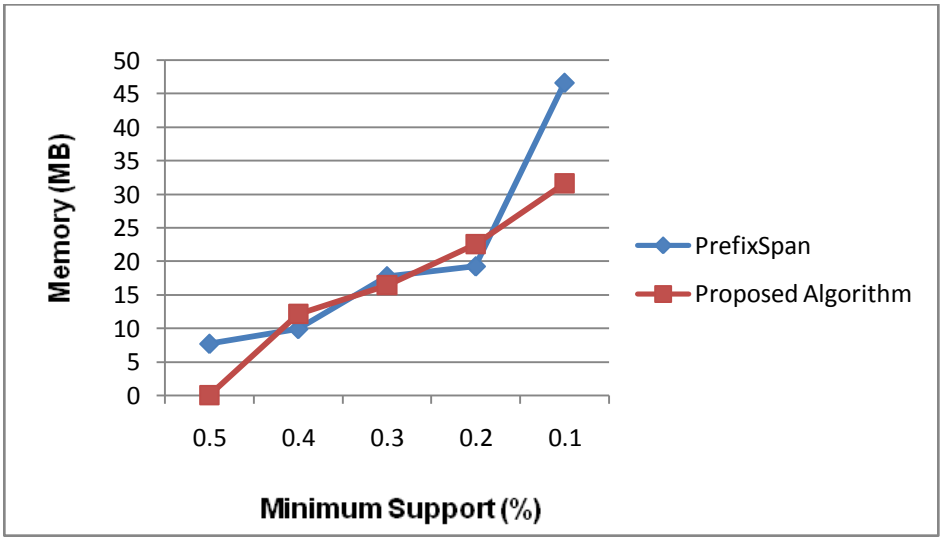

Fig 4.2: Memory Comparison on *BMS-WebView-1* data set

4.3.3 Runtime Comparison

In this section, we compare the runtime between proposed pattern mining algorithm and Prefix Span. The results are illustrated in Fig. 4.3(also table 4.3). The *x*-axis in each graph shows the change of support threshold value in decreasing order and *y*-axis indicates the overall runtime for each algorithms. The graph in Fig. 4.3 demonstrate that the runtime of proposed algorithm is not stable with respect to the change of support threshold, and it is higher than that of Prefix span algorithms in all datasets.

Support (%)	Proposed Algorithm	Prefix Span
.1	77.76	3.46
.2	15.78	0.98
.3	1.82	0.47
.4	0.79	0.23
.5	0.14	0.12

Table 4.3: Time Comparison on *BMS-WebView-1* data set

`

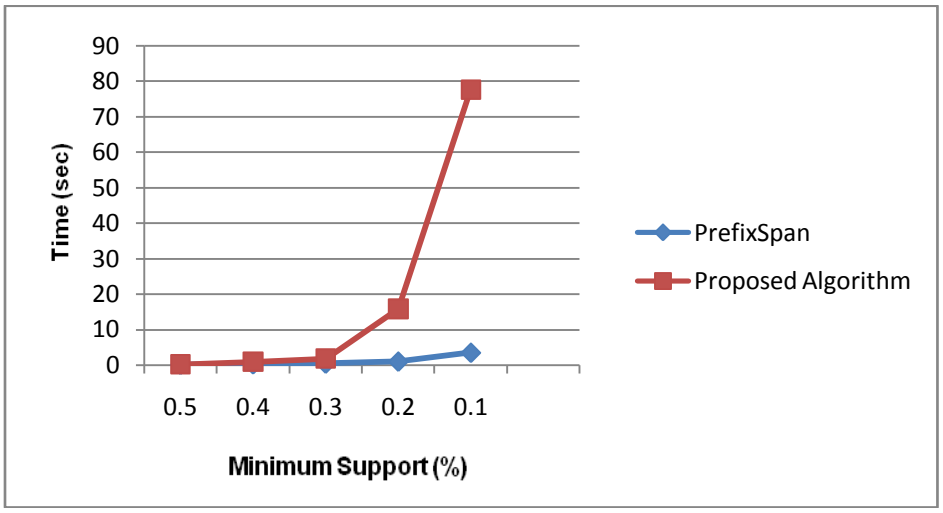

Fig 4.3: Time Comparison on *BMS-WebView-1* data set

4.4 Summary

In this chapter we introduced the environmental setup which we used while making the experimental results. In addition to this we also explain the types of dataset which we used and explain them in detail. Finally we mentioned all the analysis and experimental results that we have got and found that our proposed algorithm is both number of patterns and memory efficient in comparison to earlier algorithms.

`

Chapter 5: Conclusion & Future Scope

5.1 Conclusion

Previous traditional algorithms, can only mine closed frequent sequences of single items. In this thesis, a new technique for frequent closed pattern mining of multiple data itemsets is proposed. We have analyzed many of the earlier algorithms like Apriori, Prefix span, BIDE etc. and found that either they have large number of patterns or they take large amount of memory. So we have proposed a new technique which can check sequence closure scheme in both forward and backward direction.The best part of our proposed algorithm is that only one scan is required to construct the database and also it is memory efficient. It can also support the incremental nature of the database.

We have conducted experiments to compare the performance and efficiency of proposed algorithm with the Prefix span algorithm using real data sets. Experimental results clearly show that the proposed algorithm is highly efficient in terms of both memory and number of patterns.

The performance of the proposed algorithm is compared with the earlier algorithms using various real data sets. Experimental results clearly show that the proposed algorithm is highly efficient in terms of both memory and number of patterns.

`

5.2 Future Scope

a) Using our proposed algorithm, a number of frequent closed pattern are generated depending on the support but it does not give any information about interesting rules. So our work can be extended to give better interesting rules.

b) These days a lot more research is going on Web mining as there are large amount of data available in incremental web logs. So our algorithm can be extended to mine the patterns form the web usage data more efficiently.

c) Our proposed algorithm can become more efficient for incremental data mining.

d) Our proposed algorithm is not better in execution time. So the work can extend in terms of make it efficient in execution time.

In general, we feel that as a young research field in data mining, frequent pattern mining has achieved tremendous progress and claimed a good set of applications. However, in-depth research is still needed on several critical issues so that the field may have its long lasting and deep impact in data mining applications.

`

REFERNCES

[1]. R. Srikant , and R. Agrawal, "Mining Sequential Patterns: Generalizations and performance improvements", In *Proc. 5th Int'l Conference Extending Database Technology (EDBT)*, Avignon, France, March 1996.

[2] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth", *Proc. 2001 Int. Conf. on Data Engineering (ICDE'01)*, Heidelberg, Germany, April 2001.

[3] Jianyong Wang, Jiawei Han, "BIDE: Efficient Mining of Frequent Closed Sequences" ICDE 2004: 79-90.

[4]U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.

[5] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of International Conference on Data Engineering*, pages 3–14, 1995.

[6] T. L. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of International Conference on Intelligent Systems for Molecular Biology*, pages 28–36, 1994.

[7] E. Eskin and P. Pevzner. Finding composite regulatory patterns in dna se- quences. In *Proceedings of International Conference on Intelligent Systems for Molecular Biology*, pages 354–363, 2002.

[8] R. Agrawal and E. Wimmers. A framework for expressing and combining preferences. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 297–306, 2000.

[9] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick. Sequential pattern mining using a bitmap representation. In *Proceedings of ACM SIGKDD International Conference on*

`

Knowledge Discovery and Data Mining, pages 429–435, 2002.

[10] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu.Mining sequential patterns by pattern-growth: The prefixspan-approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424 – 1440, November 2004.

[11] Leung,C.; Khan,I.; and Hoque,T., CanTree (2005). A Tree Structure for Efficient Incremental Mining of Frequent Patterns Proceedings of the Fifth IEEE International Conference on Data Mining.

[12] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In VLDB'94, Santiago, Chile, Sept. 1994.

[13] M. Zaki, SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Learning, 42:31-60, Kluwer Academic Pulishers, 2001.

[14] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick, Sequential PAttern Mining using a Bitmap Representation. In SIGKDD'02, Edmonton, Canada, July 2002.

[15] N. Pasquier, Y. Bastide, R. Taouil and L. Lakhal, Discoving frequent closed itemsets for association rules. In ICDT'99, Jerusalem, Israel, Jan. 1999.

[16] J. Pei, J. Han, and R. Mao, CLOSET: An efficient algorithm for mining frequent closed itemsets . In DMKD'01 workshop, Dallas, TX, May 2001.

[17] M. Zaki, and C. Hsiao, CHARM: An efficient algorithm for closed itemset mining. In SDM'02, Arlington, VA, April 2002.

[18] J. Han, J. Wang, Y. Lu, and P. Tzvetkov, Mining Top- K Frequent Closed Patterns without Minimum Support. In ICDM'02, Maebashi, Japan, Dec. 2002.

[19] X. Yan, J. Han, and R. Afshar, CloSpan: Mining Closed Sequential Patterns in Large Databases. In SDM'03, San Francisco, CA, May 2003.

`

[20] “CAKE – Classifying, Associating & Knowledge Discovery An Approach for Distributed Data Mining (DDM) Using Parallel Data Mining Agents (PADMAs)”,Danish Khan, 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology.

[21] Yan Chen1, Ming Yang2, Lin Zhang3,” General Data Mining Model System Based on Sample Data Division”, 2009 Second International Symposium on Knowledge Acquisition and Modeling.

[22] Mouhib Al-Noukari, Wael Al-Hussan, Using Data Mining Techniques for Predicting Future Car market Demand”,DCX Case Study.

[23] Agrawal, R. and Srikant, R. “Fast algorithms for mining association rules.” VLDB-94, 1994.

[24] http://en.wikipedia.org/wiki/Clustering

[25] Kumar,Santosh,Rukmani,K.V.(2010) “Implementation of Web Usage Mining Using APRIORI and FP Growth Algorithms” International Journal of Advanced Networking and Applications.1 (6). Pp. 400-404.

[26] http://www.ecn.purdue.edu/KDDCUP/data/BMS-WebView-1.dat.gz

[27] "Real world performance of association rule algorithms" by Zheng, Kohavi and Mason. Available at http://robotics.stanford.edu/users/ronnyk/realWorldAssoc.pdf.

[28] Kohavi, R., Brodley, C.E., Frasca, B., Mason, L., and Zheng, Z. KDD-Cup 2000 Organizers’ Report: Peeling the Onion, SIGKDD Exploration 2(2), 2000, 86-93.

[29] Technical Report, CAIS, Nanyang Technological University, Singapore, No. 2003118 , 2003.

[30] Mining Sequential Patterns Rakesh Agrawal Ramakrishnan Srikant_IBM Almaden Research Center 650 Harry Road, San Jose, CA 95120.

`

[31] Adriaans, P., Zantinge, D.: Data Mining. Addison Wesley Longman, Harlow, England (1996) 1-10.

[32] Supaporn Erjongmanee, gtg730d@mail.gatech.edu, Chuanyi Ji, jic@ece.gatech.edu, Georgia Institute of Technology Atlanta, GA 30332," Network Service Disruption upon Natural Disaster: Inference Using Sensory Measurements and Human Inputs",year:2006.

[33] Ryan M. McCabe; Gediminas Adomavicius, PhD; Paul E. Johnson, PhD; Gregory Ramsey; Emily Rund; William A. Rush, PhD; Patrick J. O'Connor, MD, MPH; JoAnn Sperl-Hillen, MD," Using Data Mining to Predict Errors in Chronic Disease Care".

[34] Ayhan Demiriz E-Business Department, Verizon Inc., 919 Hidden Ridge, Irving, TX 75038 E-mail:ayhan.demiriz@verizon.com," A Parallel Sequence Mining Algorithm to Analyze Web Log Data".

[35] Data Mining Applications in Healthcare *Hian Chye Koh and Gerald Tan.*

[36] http://en.wikipedia.org/wiki/Clickstream

[37] *Jiawei Han and Micheline Kamber* *"Data Mining: Concepts and Techniques",2nd ed.*