Volume 5, Issue 5, May 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Comparison of Efficient Algorithms for Sequence Generation in Data Mining

Mohammad Shahnawaz Nasir
Research Scholar,
University Department of Mathematics,
Magadh University, Bodh-Gaya, India

Dr. R B S Yadav
Professor and Head,
University Department of Mathematics,
Magadh University, Bodh-Gaya, India

Abstract - Data mining is the method or the movement of analyzing data from different perspectives and summarizing it into useful information. There are several major data mining techniques that have been developed and are used in the data mining projects which include association, classification, clustering, sequential patterns, prediction and decision tree. Among different tasks in data mining, sequential pattern mining is one of the most important tasks. Sequential pattern mining involves the mining of the subsequences that appear frequently in a set of sequences. It has a variety of applications in several domains such as the analysis of customer purchase patterns, protein sequence analysis, DNA analysis, gene sequence analysis, web access patterns, seismologic data and weather observations. Various models and algorithms have been developed for the efficient mining of sequential patterns in large amount of data. This research paper analyzes the efficiency of four sequence generation algorithms namely GSP, SPADE, PrefixSpan and FreSpan on a retail dataset by applying various performance factors. From the experimental results, it is observed that the PrefixSpan algorithm is more efficient than other three algorithms.

Keywords - Sequential Pattern, GSP, SPADE, PrefixSpan, candidate generation, minimum_support, projection based.

I. INTRODUCTION

Data Mining is the discipline of finding novel remarkable patterns and relationships in vast quantity of data. Data mining technique is not developed only for a particular industry. Data Mining is considered to be very important for almost all software based applications [1]. It consists of effective techniques that help to bring out the hidden knowledge in huge volume of data. The major issue of data mining in the recent years has been focused on mining sequential patterns in a set of data sequence. The major assignment of sequential pattern mining is to determine the complete set of sequential patterns in a given sequence database with minimum user defined minimum support. Given a set of sequences and the user-specified minimum support threshold, the sequential pattern mining finds all frequent subsequences that are it identifies the subsequences whose occurrence frequency in the set of sequences is not less than minimum_support threshold [2].

Sequential Pattern Mining (SPM), [3] which discovers frequent subsequences as patterns in a sequence transactional database, is one of the significant research areas in the field of data mining. The sequence pattern mining has been generally used in many applications, including the analysis or study the customer purchase patterns in business organizations or to analyzed the web access patterns which has been distributed on multiple servers in web usage mining. Additionally, SPM is used to analyze the processes of scientific experiments, the amino acid mutation patterns in computational biology, natural disasters, and disease treatments and so on. Moreover, it plays an important role in huge databases also called as “big data”, which contains millions of records. Big data is a collection of data sets that are so huge in size and hard to process them by using on-hand database management tools or customary data processing applications. The major challenges that we are facing while dealing with big data are the capturing of data, storage, search, sharing, transfer, analysis and visualization. Searching frequent patterns plays an effective role in mining associations, correlations, and many other interesting relationships among big data.

Sequential pattern mining was first introduced by Agrawal and Srikant [2]. “Given a set of sequences, where each sequence consists of a list of events (or elements) and each event consists of a set of items, and given a user-specified minimum support threshold of min_sup, sequential pattern mining finds all frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than min_sup”. Sequential pattern mining algorithm solves the variety of problems that are associated with finding or searching the presence of frequent sequences in the specific database [4].

Many algorithms are proposed for sequential pattern mining [5]. In core terms the sequential pattern mining algorithms are classified into two different ways [6]:

(1) The way in which candidate sequences are generated and stored. The main objective is to reduce the I/O cost by reducing or minimizing the number of candidate sequences.

(2) The way in which support is counted and how candidate sequences are tested for frequency. The main purpose of this way is to remove that kind of database which has been generated for counting purposes only. On the basis of above said ways sequential pattern mining can be divided generally into two parts:

- Apriori based (GSP, SPADE, SPAM)
- Pattern growth based (FreeSpan, PrefixSpan)

In this paper, comparison is made between GSP and SPADE from Apriori-Based algorithm and FreeSpan and PrefixSpan from pattern growth approach. The study shows that PrefixSpan outperforms better than GSP, SPADE and FreeSpan algorithm for very large dataset.

The rest of the paper is organized as follows: Section-II gives the review of literature. Section-III deals with the problem objective and the proposed methodology. Section-IV discusses the GSP, SPADE, PrefixSpan and FreeSpan algorithms. Performance analysis and experimental results are presented in Section-V and conclusions are given in SectionVI.

II. LITERATURE REVIEW

Sequential pattern mining is computationally challenging because such mining may generate and/or test a combinatorial explosive number of intermediate sequence. Many novel algorithms are proposed such as Apriori, AprioriALL, GSP, SPADE, FreeSpan, SPAM and PrefixSpan.

J. Yang, W. Wang, P.S. Yu, and J. Han, [7] had performed a theoretical and simulation study on various sequential pattern mining algorithms. They proposed that PrefixSpan is an efficient pattern growth method because it outperforms GSP, FreeSpan and SPADE. They showed that the PrefixSpan Algorithm is more efficient with respect to running time, space utilization and scalability than Apriori based algorithms. Most of the existing SPM algorithms work on objective measures Support and Confidence. Their experiments showed that the percentage reduction of rule generation is high in case of interestingness measures lift. They also explained that use of interestingness measures can lead to make the pattern more interesting and can lead to indentify emerging patterns.

Mahdi Esmaeili and Fazekas Gabor [8] theoretically have shown three types of sequential patterns and some of their properties. These models fall into three classes are called periodic pattern, approximate pattern and statistically pattern. Periodicity can be full periodicity or partial periodicity. In full periodicity method, every time point contributes to the cyclic behavior of a time series. In contrast some time points in partial periodicity contribute to the cyclic behavior of a time series. This model of pattern is so rigid. The information gain can be used as a new metric that help us to discover the surprising patterns.

Huan-Jyh Shyur, Chichang Jou1, Keng Chang [9] observed the performance evaluation trend in “BMS-Webview1 dataset and Toxin-Snake dataset and showed that the SPAM method performs much better and has a better scalability than PrefixSpan in terms of execution time while in terms of memory usage, the method clearly indicates that SPAM has stable memory usage than PrefixSpan for all minimum support values. PrefixSpan algorithm is implemented with pseudoprojection technique and still by observing the performance evaluation trend it clearly shows that SPAM can be faster on sparse and dense datasets, also the memory consumption is stable as compared to PrefixSpan which is contradictory to the traditional standpoint. SPAM it generally consumes more memory than PrefixSpan and SPAM is faster on dense datasets with long patterns and less efficient on other dataset and also it consumes more memory.

Jian Pei, et al., [10] have performed a logical study on the mining of sequential patterns in Gazelle data set from Blue Martini and a pattern-growth approach had been proposed for the efficient and scalable mining of sequential patterns. Instead of refinement of sequence patterns like in the apriori-like and also instead of candidate generation-and-test approach such as in GSP, a divideand-conquer approach called the pattern-growth approach, PrefixSpan is promoted which proves to be an efficient pattern-growth algorithm for mining frequent patterns without candidate generation. PrefixSpan recursively projects a sequence database into a set of smaller projected sequence databases and grows sequential patterns in each projected database by exploring only locally frequent fragments. The entire set of sequential patterns is mined and substantially reduces the efforts of candidate subsequence generation.

III. PROBLEM OBJECTIVE AND METHODOLOGY

a. Definition

Let $I = \{i_1; i_{2 \ldots\ldots,} i_n\}$ be the set of all items. We consider an itemset as a subset of items. A sequence (event) s is set of itemsets and ordered it according to time-stamp associated with them. The sequence s is denoted as $<s_1, s_2, \ldots\ldots, s_l>$, where s_j is an itemset of s. S_j is also called an element of the sequence, and denoted as $(x_{1,} x_2 \ldots . x_m)$ where x_k is an item. An item can take place at most once in an itemset but can present multiple times in various itemsets in a sequence.

The number of instances of items in a sequence is called the length of the sequence. A sequence with length l is called an l-sequence. A sequence $\alpha=<a_1\ a_2 \ldots a_n>$ and $\beta=<b_1\ b2 \ldots b_m>$ where α is a subsequence of β, and β is a supersequence of α denoted as $\alpha \subseteq \beta$, if there exist integers $1 \leq j_1 < j_2 < \ldots < j_n \leq m$ such that $a_1 \subseteq b_{j1}$, $a_2 \subseteq b_{j2}, \ldots, a_n \subseteq b_{jn}$.

b. Problem Statement

Given a sequence database and the minimum_support threshold value, the charge of sequential pattern mining is to find the complete set of sequential patterns in the database. Four techniques namely GSP, SPADE, PrefixSpan and FreeSpan are used for generating sequences and the performance of these algorithms are analyzed and compared for finding the efficient technique.

As per data mining theory, sequential pattern mining algorithm can be generally divided into three groups: Apriori based (GSP, SPADE, SPAM), pattern growth (FreeSpan, PrefixSpan) as shown in Figure1.

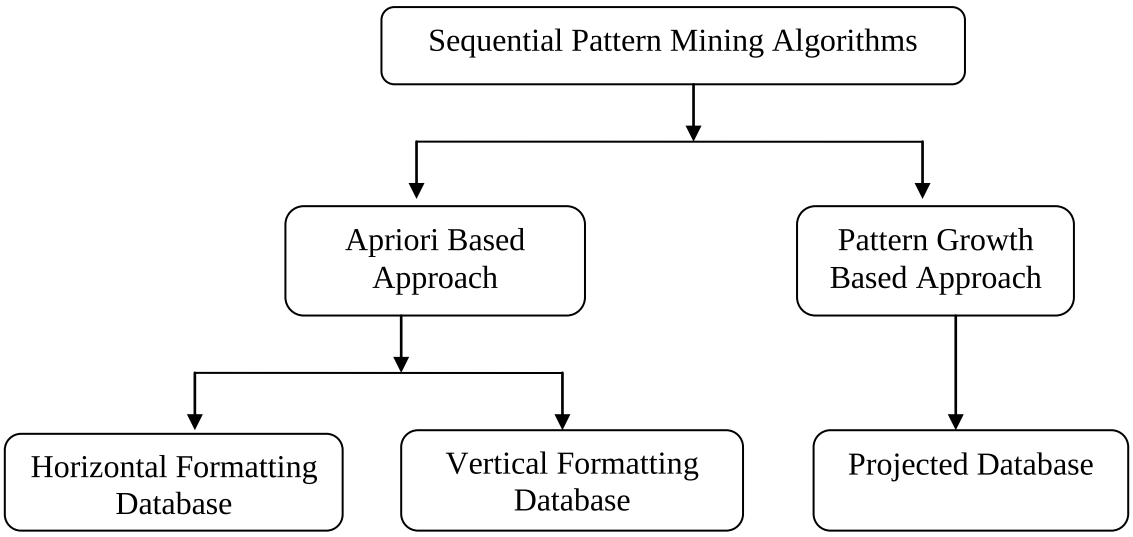
Sequential Pattern Mining Algorithms
Apriori Based Approach
Pattern Growth Based Approach
Horizontal Formatting Database
Vertical Formatting Database
Projected Database

Fig.1 Classification of Apriori based mining algorithm

IV. ALGORITHMIC APPROACHES TO THE TASK OF SEQUENCE MINING

When developing an algorithm to the task of sequence mining, the idea is to make it more efficient in terms of memory requirements and to reduce as much as possible the response time. That will imply the use of appropriated data structures and the use of old and novel algorithmic approaches to perform this task.

a. GSP (Generalized Sequence Patterns) Algorithm

GSP (Generalized Sequential Pattern Mining) algorithm is described by Agrawal and Shrikant [2] which is based on Apriori principle, mines sequential patterns by using a candidate subsequence generation-and-test approach. This algorithm does not require finding all the frequent itemsets. The algorithm works on the following principles (a) placing bounds on the time separation between adjacent elements in a pattern, (b) allowing the items included in the pattern element to span a transaction set within a time window specified by user, (c) permitting the pattern discovery in different level of a taxonomy defined by user.

This algorithm does the same work as AprioriAll algorithm but it is earlier than the AprioriAll algorithm. GSP is designed for discovering generalized sequential patterns. The GSP algorithm makes multiple passes over sequence database as follows: (1) in the first pass, it finds the frequent sequences that have the minimum support. (2) At each pass, every data sequence is examined in order to update the occurrence number of the candidates contained in this sequence.

This algorithm is not a main-memory algorithm. The algorithm creates only as many as candidates as will fit in memory and the data is scanned to count the support of these candidates. Frequent sequences that are arises from these candidates are written on disk, while those candidates without minimum support are deleted. The same step is repeated until every candidate has been counted. The GSP algorithm finds all the length-1 candidates (using one database scan) and orders them by their support value ignoring whose support<min_support. Then for each level (i.e., sequences of length k) the algorithm scans the dataset to collect the support count of the each candidates sequence and generates candidates of length (k+1) sequence from length-K frequent sequences using Apriori. This step is continued until no frequent sequence or no candidates can be found. This algorithm has a very good scale up properties with respect to the number of transaction per data sequence and number of items per transaction.

The pseudo code of GSP algorithm is as follows:

```
• Obtain a sequences in form of <x> as length-1 candidates
• Find F1 (the set of length-1 sequential patterns), after a unique scan of database.
• Let k=1;

While $F_k$ is not empty do
    - Form $C_{k+1}$, the set of length - (k+1) candidates from $F_k$;
    - If $C_{k+1}$ is not empty, unique database scan, find $F_{k+1}$ (the set of length - (k+1) sequential Patterns)
    Let k=k+1;
End While
```

One of the pitfalls of the GSP is that it generates large number of candidates that is with increasing length of sequences and the number of frequent sequences that has the tendency to decrease where the number of candidates generated by GSP is still enormous. Additionally the GSP algorithm performs multiple scans of the database which also slows down the process.

b. SPADE

Spade utilizes the prefix-based equivalence classes that decompose the original problem in to smaller sub-problems that can be solved independently in main memory using simple join operations [11]. All sequences are identified in three database scans. It uses vertical representation of the database, that is each row consists of event uniquely identified by sequence id (sid for short) and event id (eid for short).

The key features of approach are as follows:

- A vertical id-list database format is used where each of the sequence is associated with a list of objects in which it occurs along with the time-stamps. All frequent sequences can be enumerated via simple temporal joins on id-lists.
- A lattice-theoretic approach is used to divide the original search space (lattice) into smaller pieces (sub-lattices) which can be then processed independently in main-memory. The approach is performed in three database scans or only a single scan with some preprocessed information thus minimizing the I/O costs.
- The problem is then decomposed by decoupling from the pattern search. Two different search strategies are proposed for enumerating the frequent sequences within each sub-lattice: breadth-first and depth-first search.

SPADE algorithm:

```
SPADE (min_sup,D):
F1={frequent items or 1-sequences};
F2={frequent 2-sequences};
E={equivalence classes[X]_1};
For all [X]_E do Enumerate-Frequent-Seq([X]);
```

SPADE minimizes the I/O costs by reducing database scans and as well as minimizes the computational costs by employing efficient methods for searching. Data-skew can occur since the vertical id-list based approach is insensitive to it.

c. PREFIXSPAN

PrefixSpan is different from the normal way of generating candidates and testing them such as GSP and SPADE. The PrefixSpan algorithm has two key features [12]:

- It is projection-based.
- The patterns are generated sequentially in the projected databases by investigating only locally frequent segments.

The PrefixSpan (Prefix-projected Sequential pattern mining) algorithm presented by Jian Pei, Jiawei Han and Helen Pinto [13] representing the pattern-growth methodology, which finds the frequent items after scanning the sequence database once. This algorithm is based on the idea of database projection and sequential pattern growth. This algorithm works in a divide-conquer way, examines only the prefix subsequences after scanning the sequence database once and then projects their corresponding postfix subsequences into projected database likewise sequential pattern are developed in each projected database by exploring only local frequent sequences. Projected database keeps on shrinking because only the suffix subsequences of a frequent prefix are projected into a projected database. In prefixspan different projections methods can be used and these are level-by-level projection, bi-level projection and pseudo projection.
In level-by-level projection stats with scanning the sequential database and take out the length-1 sequence from it. In next step, the sequential database is separated into various partitions based on the number of length-1 sequences and each partition is the projection of the sequential database that takes the corresponding length-1 sequences as prefix. The projected databases only hold the postfix of these sequences by scanning the projected database all the length-2 sequential patterns that have the parent length-1 sequential patterns as prefix can be generated. Then the projected database is partitioned again by those length-2 sequential patterns.
The same process is executed frequently until the projected database is empty or no more frequent length-k sequential patterns can be generated. The method mentioned above is called level-by-level projection, there is no candidate generation process. The cost mainly occurred in this method is the time and space used to construct projected databases. Another projection method called bi-level projection is planned to decrease the number and size of projected databases.

Algorithm of PrefixSpan

```
Input a sequence database S and the least support threshold, min_support
Call PrefixSpan(<>,0,S)
Procedure PrefixSpan (α, L, Sα)
1) Scan Sα once, find each frequent item b, such that:
        a) b can be assembled to the last element of α to form a sequential pattern; or
        b) <b> can be appended to α to form a sequential pattern.
```

2) For each frequent item b, append it to α to form a sequential pattern α ' and output α '.
3) For each α ', construct α '-projected database S α '.
4) Call PrefixSpan (α ', L+1, Sα ')

PrefixSpan outperformed the other methods mainly in three ways:

- It grows patterns without candidate generation.
- The data reduction can be performed effectively by the projections.
- The memory space utilization is approximately steady.

d. FreeSpan (Frequent Pattern Projected Sequential Pattern Mining)

FreeSpan is an algorithm proposed by Pei et al. In 2001 [14] with an objective is to decrease the generation of candidate subsequences.
It uses divide-and-conquer approach to recursively project the sequence database into projected databases while growing subsequence fragments in each projected database. This process partitions both the data and the set of frequent patterns to be tested, and confines each test being conducted to the corresponding smaller projected database. FreeSpan first scans the database, collects the support for each item, and finds the set of frequent items. Frequent items are listed in support descending order. Each projection partitions the database and limitations further testing to increasingly smaller and more manageable units. Two alternatives of database projections can be used Level-by level projection or Alternative-level projection.

V. EXPERIMENTAL RESULTS

In this section we performed a comparative study between GSP, SPADE, PrefixSpan and FreeSpan on both the real and synthetic or artificial data for the purpose of determining the effectiveness and efficiency of the PrefixSpan algorithm. All experiments were conducted on a 2 GHZ PC with 2GB main memory, running Microsoft Windows 2000 Server. Four algorithms, GSP, SPADE, PrefixSpan and FreeSpan, were implemented by using C++.with retail market data set. The data set used in this research taken from frequent item Set Mining repository. We obtained the data set from http://www.sigkdd.org/kddcup/ index.php?section =2000 &method=data considered it as a real dataset. This data set contains 29,369 sequences (i.e., customers), 35,722 sessions (i.e., transactions or events), and 87,546 page views (i.e., products or items).
In this research work, four sequential pattern mining algorithms namely GSP, SPADE, PrefixSpan and FreeSpan are used to generate the sequential patterns from the retail dataset. For this study, various threshold levels are used and their results are analyzed. The size of the transaction is 100 and the average length of the transactions is 8. The samples of sequence patterns generated by these algorithms are given in Table 1 below:

TABLE I: Sample Sequences generated by the algorithms

S.No.	*Sequences Produced*
1	*(105)*
2	*(105, 152)*
3	*(105, 225)*
4	*(105, 32)*
5	*(105, 152, 225)*
6	*(105, 152, 225, 32, 36)*

The Table 2 shows the execution time of GSP, SPADE, PrefixSpan and FreeSpan algorithms at various threshold levels for the retail data set.

TABLE II: Execution Time for GSP, SPADE, PrefixSpan and FreeSpan

Algorithm	***Execution time (in millisec)***		
	Min_sup = 5	***Min_sup = 10***	***Min_sup = 15***
GSP	*36*	*37*	*39*
SPADE	*28*	*30*	*27*
PrefixSpan	*22*	*20*	*21*
FreeSpan	*24*	*25*	*24*

The below graph shows the execution time of four different algorithms. From the experimental results, the PrefixSpan algorithm needs less execution time than GSP, SPADE and FreeSpan algorithm.
The Table 3 shows the memory space utilized by GSP, SPADE, PrefixSpan and FreeSpan algorithms at various threshold levels.

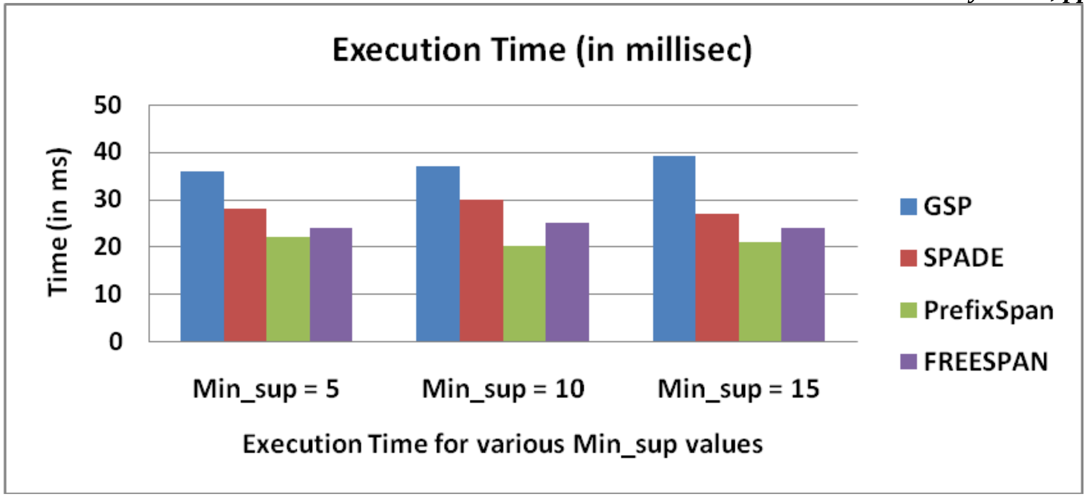

Fig.2 Execution Time: GSP, SPADE, PrefixSpan and FreeSpan

TABLE III: Memory Space Utilization

Algorithm	*Memory Space Utilization (in kb)*		
	Min_sup = 5	*Min_sup = 10*	*Min_sup = 15*
GSP	*422*	*386*	*388*
SPADE	*350*	*351*	*346*
PrefixSpan	*323*	*323*	*317*
FreeSpan	*341*	*342*	*340*

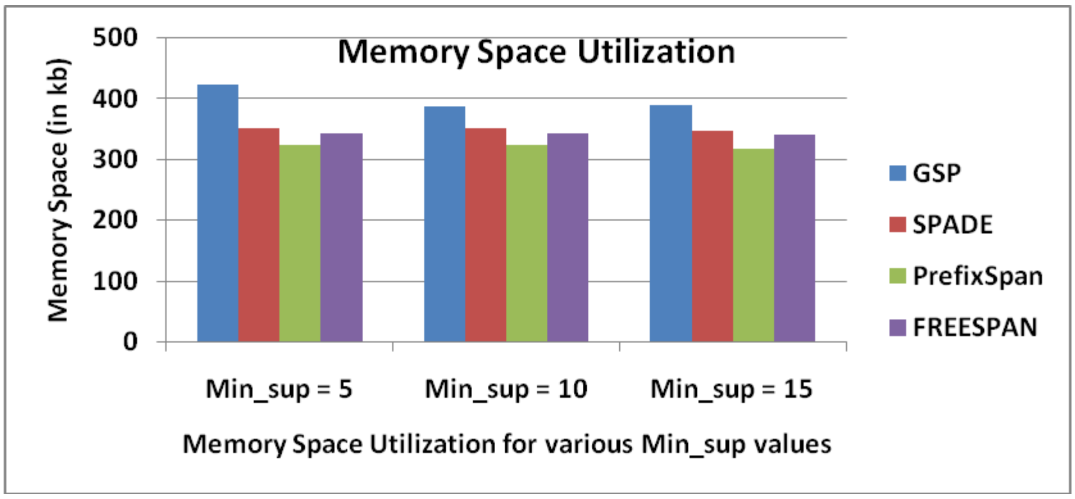

Fig.3 Memory Space Utilization: GSP, SPADE, PrefixSpan and FreeSpan

The above graph shows the memory space utilization of sequence generation algorithms. The results shows that the memory space utilized by PrefixSpan occupies less memory space compared to GSP, SPADE and FreeSpan algorithm.

VI. CONCLUSION

Sequential mining has been attracting attention in recent research in the field of data mining. Since the search space is very large and data volume is huge, it has made many problems for mining sequential patterns. In order to effectively mine the sequential patterns, efficient sequential pattern mining algorithms are needed. Among the sequential pattern algorithms GSP, SPADE, PrefixSpan and FreeSpan, PrefixSpan is an efficient pattern growth method because it outperforms the other three algorithms. It is clear that PrefixSpan Algorithm is more efficient with respect to running time, space utilization and scalability than Apriori based algorithms. Future research may involve the development of novel measures which can make the pattern more interesting and can be helpful to identify emerging patterns.

REFERENCES

[1] Zheng Zhu, “Data Mining Survey - ver 1.1009”
[2] R.Agrawal and R.Srikant, “Mining Sequential Patterns," In Proceedings of International conference on data engineering. pp. 3-14
[3] Han J., Dong G., Mortazavi -Asl B., Chen Q., Dayal U., Hsu M.C.FreeSpan: Frequent pattern projected sequential pattern mining, Proceedings 2000 Int. Conf. Knowledge Discovery and Data Mining (KDD’00), 2000, pp. 355-359.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94), pages 487–499, Santiago, Chile, Sept. 1994.

[5] Pedro Gabriel Dias Ferreira," A survey on Sequence Pattern Mining Algorithms".

[6] Nizar R. Mabroukeh and C. I. EZEIFE, ‖A Taxonomy of Sequential Pattern Mining Algorithms, ACM Computing Surveys, Vol. 43, No. 1, Article 3, Publication date: November 2010.

[7] J. Yang, W. Wang, P.S. Yu, and J. Han, "Mining Long Sequential Patterns in a Noisy Environment,"Proc. 2002 ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD '02), pp. 406-417, June 2002.

[8] Mahdi Esmaeili and Fazekas Gabor, "Finding Sequential Patterns from Large Sequence Data",IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 1, No. 1,January 2010 ISSN (Online): 1694-0784 ISSN (Print): 1694-0814

[9] Huan-Jyh Shyur, Chichang Jou1, Keng Chang, "A data mining approach to discovering reliable sequential patterns", The Journal of Systems and Software 86 (2013) 2196– 2203.

[10] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, Mei-Chun Hsu, "Mining Sequential Patterns by Pattern-Growth: The PrefixSpan approach", IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 10, October 2004

[11] M. Zaki, "SPADE: An efficient algorithm for mining frequent sequences, Machine Learning, 2001.

[12] Manan Parikh, Bharat Chaudhari and Chetna Chand, "A Comparative Study of Sequential Pattern Mining Algorithms", International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 2, Issue 2, February 2013 ISSN 2319 – 4847

[13] J. Pei, J. Han, B. Mortazavi-Asi, H. Pinto, "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth", ICDE'01, 2001.

[14] M.J. Zaki. Scalable algorithms for Association Mining. IEEE Transactions on Knowledge and Data Engineering. 12(3), 372-390, 2000.