Comparative Study of Various Sequential Pattern Mining Algorithms

Ravi Kumar Verma

Assistant Professor, Department of Computer Science
Parthivi College of Engineering & Management, Bhilai
Raviverma07[at]gmail.com

Abstract*: In successive pattern mining represents a very important category of knowledge mining issues with big selection of applications. It's one in every of the terribly difficult issues as a result of it deals with the careful scanning of a combinatorial sizable amount of attainable subsequence patterns. Broadly speaking successive pattern Ming algorithms are often classified into 3 sorts specifically Apriori based mostly approaches, Pattern growth algorithms and early pruning algorithms. These algorithms have more classification and extensions. Elaborate rationalization of every formula alongside its necessary options, pseudo code, blessings and drawbacks is given within the future sections of the paper. At the top a comparative analysis of all the algorithms with their supporting options is given within the variety of a table. This paper tries to complement the information and understanding of varied approaches of sequential pattern mining.*

Keywords : Basic Apriori, GSP, SPADE, PrefixSpan, FreeSpan, LAPIN**,** Early pruning.

1. Introduction

Sequential pattern mining is a significant topic of data mining with wide range of applications. It deals with extracting statistically useful patterns between data which occurs sequentially with a specific order. Sequential pattern mining is considered as a special case of structured data mining. It is considered to be a complex problem because in it a combinatorial explosive number of intermediate subsequences are generated. Sequential pattern mining is used in several domains [2] such as in business organizations to study customer behaviors, in web usage mining to mine several web logs distributed on multiple servers. The sequential pattern mining problem can be described as follows [1]: Consider a given a set of data-sequences, as the input wherein each data-sequence is a list of transactions such that each transaction contains a set of items. Given a user-specified minimum support threshold, then sequential pattern mining is applied to find out all frequently occurring subsequences whose ratios of occurrence exceed the minimum support threshold in the sequence database. Records are stored in a sequence database such that all the records are sequences of ordered events, with or without concrete notions of time [5]. An example sequence database is retail customer transactions showing the sequence or collection of products they purchased on weekly or monthly basis. A sequential pattern-mining algorithm can be

2. Extend the mining of sequential patterns to other time-related patterns [2]. This strategy focuses on finding other patterns in time-related databases such as finding frequent traversal patterns in a web log, cyclic patterns in a time-stamped transaction database etc.

2. Literature Review

All As discussed earlier, sequential pattern is a set of itemsets arranged in sequence database occurring sequentially in a specific order [2]. A sequence database is a collection of ordered elements or events with or without a view of time. Each itemset contains a set of items which appear together in the same transaction and thus have the same session time value [2,1]. Whereas association rules indicate intra-transaction relationships, sequential patterns represents inter transaction relationships i.e. the relationship between transactions. Given two sequences α=<a1 a2…an> and β=<b1 b2…bm> where α is called a subsequence of β, denoted as $\alpha \subseteq \beta$ if there exist integers $1 \le j1 < j2 < \cdots < jn \le m$ such that $a1 \subseteq bj1$, $a2 \subseteq bj2, \cdots, an \subseteq bjn$. Here if α and β have the following sequences α=<(xy), t> and β=< (xyz), (zt)>, β is denoted as a super sequence of α [2,6]. In addition to the discovery of recurrent itemsets, sequential pattern mining requires the arrangement of those itemsets in a sequence. Suppose Is denotes powerset of a set of items denoted by Cs. Let s represents minimum support threshold for mining the database and let m =|C|. Then aim of mining frequently occurring itemset is to discover recurrent itemsets among | Is | different possible itemsets as represented in equation (i) below [6]:

$$I = \sum_{s=1} - 1 = 2 - 1 \qquad \text{(i)}$$

Now suppose that the database has sequences with at most p itemsets and each itemset has at most single item such that there are m^p possible different sequences the different variable length sequences are given by equation (ii) as below [6]:

$$= \frac{+1}{} \quad \text{(ii)}$$
$$\bar{I} \qquad -1$$

Now let S_n be the possible frequent sequences with n itemsets then value of S_n is represented in equation (iii) as below [6]:

$$= I_s = 2 - 1 \quad \text{(iii)}$$

In general the existing S_T sequences are given by the equation (iv) as shown below [6]:

$$= \bar{I} \; 2 - 1 = \frac{2^{+1-2} - 1^{-1}}{2^{-2}} = 2 \quad \text{(iv)}$$

Consider the example of a customer sequence [7] where each transaction made corresponding to a set of items is ordered by increasing transaction time. Table 1 shows the database corresponding to consumer transactions sorted by consumerID and Transaction time [7,1] as shown below:

Table 1 : Table showing database for consumer transactions

ConsumerID	**TransactionTime**	**ItemsBought**
1	November 25 '13 November 30 '13	30 90
2	November 10 '13 November 15 '13 November 20 '13	10,20 30 40,60,70
3	November 25 '13	30,50,70
4	November 25 '13 November 30 '13 December 25 '13	30 40,70 90
5	November 12 '13	90

Next, a customer sequence is created according to time duration as shown in Table 2 that contains consumerID and Consumer sequence for items bought [7,1] :

Table 2: Table showing customer sequence according to time duration

ConsumerID	Consumer Sequence
1	<(30) (90)>
2	<(10 20) (30) (40 60 70)>
3	<(30 50 70)>
4	<(30) (40 70) (90)>
5	<(90)>

The support for an itemset i_t is defined as the fraction of customers who bought the items in i_t in a single transaction. An itemset having minimum support is known as a large itemset or simply litemset [7]. It is required that each itemset in a large sequence must have minimum support. In this example of customer transaction sequence database is shown in Table 1, the large itemsets so found are (30), (40), (70), (40 70) and (90). A possible mapping for this set is shown in Table 4 in order to treat litemsets as single entities. Thus two litemsets can be compared for equality in constant time and it also decreases the time neededd for checking whether a sequence is contained in a customer sequence or not [7].

Table 3. Table showing mapping of large item sets

Large Itemsets	**Mapping To**
(30)	1
(40)	2
(70)	3
(40 70)	4
(90)	5

3. Classification Of Sequential Pattern Mining Algorithms

The three main categories of Sequential pattern mining algorithms that have been proposed are discussed as follows [1, 5]:

1. Apriori Based algorithms such as GSP, SPADE, SPAM algorithms
2. Pattern Growth algorithms such as FreeSpan and PrefixSpan
3. Early Pruning algorithms such as LAPIN-SPAM. Some other algorithms are hybrids of these techniques.

3.1 Apriori Based algorithms

3.1.1 Basic Apriori Algorithm

Apriori algorithm is classically used over transactional databases for mining frequently occurring itemsets and association rule generation. It is a difficult task to develop efficient and scalable methods for mining frequent itemsets in a large transaction database because normally there are a large number of distinct single items in a

typical transaction database and furthermore their combinations may result in a very huge number of itemsets [3]. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets provided that those item sets occur sufficiently often in the database. An downward closure property known as Apriori is observed among frequent k-itemsets such that: A k-itemset is frequent if all of its sub-itemsets are frequent [3,8]. Thus frequent itemsets can be extracted by first examining the database to find the frequent 1-itemsets, then the frequent 1-itemsets can be used to generate candidate frequent 2-itemsets and then database is checked to obtain the frequent 2-itemsets. This process repeats till no more frequent k-itemsets can be generated for some k. This forms the fundamental of the Apriori algorithm [3].The frequent item sets so extracted by Apriori algorithm can be used to generate association rules which emphasize the general trends in the database and are applicable in variety of domains such as market basket analysis. The mining process was decomposed with five steps [7]:

a. **Sort step:** In this step the transactional database is sorted according the customer-id which is regarded as primary key and transaction time is taken as minor key.

b. **L-itemset step**: In this step the main aim is to obtain the large 1-itemsets from the sorted database, based on the support threshold. Here first the set of all litemsets L is extracted. At the same time the set of all large 1-sequences are found as given by [7,2]: {(l) | l € L}

c. **Transformation step**: In this step the sequences are replaced by the large itemsets they contain. Then all the large itemsets are mapped to integer series for efiicient mining [7, 2]. If in case a transaction does not have any litemset then it is not retained in the transformed sequence [7].

d. **Sequence step**: In this step all frequent sequential patterns are generated from the transformed sequential database.

e. **Maximal step**: This step prunes the sequential patterns that are contained in large sequential patterns since concern is only with maximum sequential patterns so focus is to find the maximal sequences among the set of large sequences. Sometimes maximal step is combined with the sequence phase to decrease the time wasted in counting non-maximal sequences [7, 2].

The pseudocode of Apriori algorithm is given as follows [8]:

Ck: Candidate itemset of size k

Lk: frequent itemset of size k

L_1 = {Large 1-itemsets}; // Outcome of litemset phase **for**(k= 2; L_{k-1}=∅; k++) **do begin**

C_k = New candidates generated from L_{k-1};
foreach transaction ***t*** in database do

Increment the count of all candidates in C_k that are contained in t

end

L_k = candidates in C_k with minimum support

end

return ;

Various methods have been proposed for improving the efficiency of Apriori algorithm. Some of the proposed work includes sampling approach, dynamic itemset counting, incremental mining, parallel and distributed mining [2]. A point of concern in Apriori algorithm is that in some cases, the size of candidate sets using the Apriori principle is significantly reduced and lead to two main problems [2, 5]:

a. **Generate-and-test:** This is the feature of early sequential pattern mining algorithms. It proposes use of exhaustive join operators such that the pattern simply grows one item at a time and tested for the minimum support. This leads to an inefficient pruning and generates an explosive number of candidate sequences and thus taking up a lot of memory [5].

a. **Scanning database repeatedly:** This feature suggests multiple scanning of the original database to determine if the list of generated candidate sequences is frequent or not. It is a very detrimental characteristic as it requires a lot of processing time and I/O cost. To overcome this drawback database should be scanned only once or twice to generate a temporary data structure containing necessary information used in mining [5].

3.1.2 GSP Mining Algorithm

Generalized Sequential Patterns (GSP) also follows Apriori-based approach of sequential pattern mining. Algorithm uses the downward-closure property of sequential patterns as discussed in previous section and follows a multiple pass candidate generate-and-test approach [3]. GSP follows the principle of candidate generation and test. The algorithm starts by detecting the frequent 1-sequences and then generating the collection of frequent (*k+1*)-sequences derived from the set of frequent *k*-sequences usually known as candidates [6]. The frequent *k*-sequences (*k* candidates) so generated use the frequent (*k-1*)-sequences and this reduces the number of sequences to consider at each moment [6]. GSP also includes time constraints, a sliding time window, and user-defined taxonomies [3]. The features of GSP algorithm are as follows [2]:

a. The algorithm allows inserting bounds on the time separation between contiguous elements in a pattern.

b. It allows the items present in the pattern element to span a transaction set within a time window specified by user.

c. It allows discovery of frequent patterns in
d. different levels as desired by user.

e. It also enables for discovering generalized sequential patterns.

The GSP algorithm makes multiple passes over sequence database as per the following procedure [2, 6]:

1. In the first pass, from a given set of frequent n-1 patterns, the candidate set for next generation are generated and based on the thresholds it finds the

frequent sequences that have the minimum support.

2. The frequent patterns in the current set are considered for generating the next candidate sequence.
3. Every data sequence is examined during each phase so as to update the occurrence number of the candidates contained in this sequence.
4. It removes those candidates that have some non-frequent maximal subsequence.

The pseudo code of GSP algorithm is as follows [2]:

1. Let F_1 denotes the set of frequent 1-sequence after database scan
2. Let k=1;
3. Do while F(k)!= Null;
4. Generate candidate sets C_{k+1} set of candidate k+1 sequences
5. If C_{k+1} is not empty, find F_{k+1} i.e. the set of length-(k+1) sequential patterns
6. k= k+1;
7. End do

GSP Algorithm suffers from the following drawbacks [2]:

- Multiple scans of the database are needed for generation of a huge set of candidate sequences

The algorithm is not suitable for mining long sequential patterns because of the expensive number of short patterns for the mined pattern length.

3.1.3 SPADE Mining Algorithm

The Sequential PAttern Discovery Algorithm Using Equivalence classes also called SPADE algorithm was first introduced by Mohammed J. Zaki in [4], for discovering the set of all frequent sequences. The main characteristics of SPADE algorithm are described as below [4]:

I. The algorithm proposes the usage of vertical id-list database format with the help of which each sequence can be associated with a list of objects in which it occurs in conjunction with the related time-stamps [11].

II. On implementing simple temporal joins on id-lists frequent sequences are enumerated [11].

III. A lattice-theoretic approach is used to divide the original lattice into smaller sub-lattices such that each one is processed independently in main-memory.

IV. In order to minimizes the I/O costs the algorithm requires up to three database scans or alternatively one scan with some pre-processed information [11].

V. The problem decomposition task is separated from pattern search process.

VI. Two different search strategies have been defined in order to find and list frequent sequences in each sublattice, namely breadth-first search (BFS) and depth-first search (DFS) [11].

The SPADE algorithm can be described as follows [4, 2]:

Let D denote ID-List of sequences

Let ms denote Minimum Support

Let F_S denote set of sequences

SPADE (ms, D)

1. Let F_1 = Frequent 1-Sequences
2. Let F_2 = Frequent 2-Sequences
3. Determine E as set of Equivalence classes for all 1-Sequences $[Q]_{\Theta 1}$
4. For each Q € E do

Find frequent sequences F_S

Advantages of SPADE algorithm can be enumerated as follows [2, 4]:

a) SPADE outperforms GSP since it is about twice as fast as GSP because it uses a support counting method based on the id list structure which is more efficient method.

b) SPADE scales linearly with respect to the database size, number of sequences and other database parameters.

c) SPADE decreases I/O costs by reducing database scans.

d) SPADE reduces computational costs by using efficient search schemes.

e) The vertical id-list based approach that SPADE follows is not sensitive to data-skewing.

3.2 Pattern Growth Algorithms

Please The Pattern-growth methods represent a new approach for handling problems related to sequential pattern mining by providing solution to the problem of generate-and-test. The search space partitioning is an important feature of pattern-growth algorithms [5]. The algorithm avoids repeatedly scanning the entire database for generating and testing large sets of candidate sequences. Instead it proposes to recursively project a sequence database into a set of smaller databases and locally mining frequent patterns in each projected database [11]. The idea is to avoid the candidate generation step concentrate the search on a smaller and limited portion of the database [6, 5]. The initial pattern growth algorithms used projected databases and two types of database projections can be used either Level-by-level projection or Alternative-level projection. The two popular projection-based sequential pattern mining methods are: FreeSpan [12] and an improved method PrefixSpan [13]. Both methods generate projected databases with difference in the database projection criteria. While FreeSpan uses current set of frequent patterns not in particular growth direction (ordering), to create projected databases, PrefixSpan projects databases by growing frequent prefixes [6]. PrefixSpan is a better pattern-growth method that focuses on constructing patterns recursively and limiting the search to projected databases.

3.2.1 FreeSpan Algorithm

FreeSpan algorithm has been proposed with the intend to reduce the generation of candidate subsequences by using projected databases for mining frequent patterns. In order to generate subsequence fragments in each projected database

FreeSpan uses frequent items to project sequence databases into a set of smaller projected databases recursively using the currently mined frequent sets [2]. In FreeSpan database is scanned three times regardless of maximal length of the sequence. FreeSpan is efficient and faster than the GSP algorithm but the major issue of FreeSpan is to deal with projected databases [2]. The major steps in the FreeSpan algorithm can be explained as follows [12]:

Let sequence database be denoted by S_T.

Let minimum supported threshold as *ms* then

1. Scan S_T to find set of frequent items in S_T
2. Sort frequent items in above step in Freq_list in the decreasing order of frequency.
3. Execute alternative level projection mining in the following steps:
   a. Scan the database once to create a frequent item matrix.
   b. Generate length-2 sequential patterns on item-repeating patterns and projected databases.
   c. Scan database to generate item repeating patterns and projected databases.
   d. If still longer candidate patterns exist then recursively execute matrix projection mining on projected databases.

3.2.2 PrefixSpan Algorithm

PrefixSpan algorithm represents a new pattern-growth approach for mining sequential patterns. The algorithm proposes that the projection of sequence databases is based only on occurance of frequent prefixes and not by considering all the possible occurrences of frequent subsequences since any frequent subsequence can be found by growing its frequent prefix [13]. The PrefixSpan algorithm scans only the prefix subsequences and projects their corresponding postfix subsequences into the databases. Thus, sequential patterns in each projected database grow by exploration [5]. The steps of PrefixSpan algorithm are given as follows [13]:

Let sequence database be denoted by *S*.

Let minimum supported threshold as *ms* then

Let p denotes sequential pattern and len denotes the length of p.

Let $S|_p$ denotes the p projected database if $p \neq <>$, otherwise the sequence database S.

1. Scan $S|_p$ to find the set of frequent items k such that either of below two steps can take place:
   a. k can be assembled to the last element of p for forming a sequential pattern or,
   b. <k> can be appended to p to form sequential pattern.
2. For each frequent item k append it to p to generate sequential pattern p_1 and show p_1 as output.
3. For each p_1 construct p_1-projected database $S|_{p1}$ and repeat the steps.

The features of PrefixSpan can be summarized as follows:

i. No candidate sequence needs to be generated by PrefixSpan.

ii. A projected database is smaller than the original database since in it only the postfix subsequences of a frequent prefix are projected. Thus, the projected databases keep on shrinking.

In the worst case, PrefixSpan constructs a projected database for every sequential pattern.

3.3 Early Pruning Algorithms

The Early-Pruning approaches are emerging as a new class of algorithms for sequential pattern mining. These algorithms use a type of position induction concept to prune candidate sequences at early stage in the mining and the remaining algorithm represents a simple pattern growth process [5]. The concept of position induction is as follows: During mining process if an item's last position is smaller than the current prefix position then the item cannot appear behind the current prefix in the same customer sequence [5]. These algorithms use a table to record the last positions of each item in the sequence and later use this information for early candidate sequence pruning because the last position of an item can be effectively used to decide whether the item can be appended to a given prefix k-sequence or not, thus avoiding support counting and generation of candidate infrequent sequences [5]. LAPIN is an example of early-pruning algorithms. The following table shows the comparative features of the different sequential pattern mining algorithms [1, 2, 5].

Table 4. Table showing comparative features of different sequential pattern mining algorithms

Algorithm Characteristics	Apriori All	GSP	SPADE	FreeSpan	Prefix Span	LAPIN
Generate and Test	Support	Support	Support			
MultiScan of Database	Support	Support				
Candidate Sequence Pruning		Support	Support		Support	Support
Sampling and/or compression	Support					
DFS based approach			Support	Support	Support	Support
BFS based approach		Support				
Top-down search				Support	Support	
Bottom-up search		Support	Support			
Prefix growth						Support
Search Space Partitioning	Support		Support	Support	Support	Support
Database vertical projection			Support			Support
Support counting						Sup

avoidance						port
Position coded						Sup port

4. Conclusion

In this paper a very important and complex data mining problem known as Sequential pattern mining has been analyzed in details. This paper gives a brief description of the latest and popular sequential pattern mining algorithms as classified under three broad categories namely: apriori-based algorithms, pattern growth methods and early-pruning based algorithms and their subtypes and extensions. Under the Apriori based algorithms category: Basic apriori algorithm, GSP and SPADE algorithms have been described; under pattern growth approaches FreeSpan and PrefixSpan algorithms have been explained; basics of early pruning methods and its popular algorithm LAPIN have also been defined further. The paper presents a detailed explanation of their features, respective advantages and disadvantages. At the end comparative analysis of these algorithms based on their characteristic features is presented. In depth research work needs to be conducted for extending the capabilities of existing sequential mining approaches. Various considerations in this direction include distributed mining of sequences, object oriented view of sequential pattern mining, sequence regeneration and future items predictions in a given sequence using mathematical models of sequences etc.

References

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

[2] Thabet Slimani, and Amor Lazzez, Sequential Academic Publishers. Manufactured in The Netherlands zaki@cs.rpi.edu

[3] Jiawei Han · Hong Cheng · Dong Xin · Xifeng Yan, Frequent pattern mining: current status and future directions Received: 22 June 2006 / Accepted: 8 November 2006 / Published online: 27 January 2007 Springer Science+Business Media, LLC 2007

[4] Mohammed J. Zaki, SPADE: An Efficient Algorithm for Mining Frequent Sequences, Machine Learning, 42, 31–60, 2001 2001 Kluwer.

[5] NIZAR R. MABROUKEH and C. I. EZEIFE, A Taxonomy of Sequential Pattern Mining Algorithms. University of Windsor,

[6] Cláudia Antunes and Arlindo L. Oliveira Instituto Superior Técnico, Sequential Pattern Mining Algorithms: Trade-offs between Speed and Memory, INESC-ID, Department of Information Systems and Computer Science, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal

[7] Rakesh Agrawal and Ramakrishnan Srikant, Mining Sequential Patterns, IBM Almaden Research Center 650 Harry Road, San Jose, CA 95120

[8] Rakesh Agrawal Ramakrishnan Srikan, Fast Algorithms for Mining Association Rules, IBM Almaden Research Center 650 Harry Road, San Jose, CA 95120

[1] A. Bonnaccorsi, "On the Relationship between Firm Size and Export Intensity," Journal of International Business Studies, XXIII (4), pp. 605-635, 1992. (journal style)

[2] R. Caves, Multinational Enterprise and Economic Analysis, Cambridge University Press, Cambridge, 1982. (book style)

[3] M. Clerc, "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization," In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), pp. 1951-1957, 1999. (conference style)

[4] H.H. Crokell, "Specialization and International Competitiveness," in Managing the Multinational Subsidiary, H. Etemad and L. S, Sulude (eds.), Croom-Helm, London, 1986. (book chapter style)

[5] K. Deb, S. Agrawal, A. Pratab, T. Meyarivan, "A Fast Elitist Non-dominated Sorting Genetic Algorithms for Multiobjective Optimization: NSGA II," KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000. (technical report style)

[6] J. Geralds, "Sega Ends Production of Dreamcast," vnunet.com, para. 2, Jan. 31, 2001. [Online]. Available: http://nl1.vnunet.com/news/1116995. [Accessed: Sept. 12, 2004]. (General Internet site)

[7] Murat Ali Bayır, Ismail H. Toroslu and Ahmet Coşar, A Performance Comparison of Pattern Discovery Methods on Web Log Data ,Department of Computer Engineering Middle East Technical University.

[8] Behzad Mortazavi-Asl ,Discovering And Mining User Web-Page Traversal Patterns , Simon Fraser University, 1999.

[9] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen Umeshwar Dayal, and 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

[10] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu, "FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining," Proc. 2000 ACM SIGKDD Int'l Conf. Knowledge Discovery in Databases (KDD '00), pp. 355-359, Aug. 2000.

[11] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "PrefixSpan: Mining Sequential

[12] Patterns Efficiently by Prefix-Projected Pattern Growth," Proc. 2001 Int'l Conf. Data Eng. (ICDE '01), pp. 215-224, Apr. 2001