SPMLS : An Efficient Sequential Pattern Mining Algorithm with candidate Generation and Frequency Testing

M.Parimala

Senoir Lecturer, Department of Computer Applications,
M. Kumarasamy college of Engineering,
Karur, Tamilnadu,
India.
Email: pariguns@gmail.com

S.Sathiyabama
Assistant Professor of Computer Science,
Thiruvalluvar Govt Arts and Science College,
Rasipuram, Tamilnadu,
India.

Abstract - Sequential pattern mining is a fundamental and essential field of data mining because of its extensive scope of applications spanning from the forecasting the user shopping patterns, and scientific discoveries. The objective is to discover frequently appeared sequential patterns in given set of sequences. Now-a-days, many studies have contributed to the efficiency of sequential pattern mining algorithms. Most existing algorithms have verified to be effective, however, when mining long frequent sequences in database, these algorithms do not work well.

In this paper, we propose an efficient pattern mining algorithm, SPMLS, Sequential Pattern Mining on Long Sequences for mining long sequential patterns in a given database. SPMLS takes up an iterative process of candidate-generation which is followed by frequency-testing in two phases, event-wise and sequence-wise. Event-wise phase presents a new candidate pruning approach which improves the efficiency of the mining process. Sequence-wise phase integrates considerations of intra-event and inter-event constraints. Simulations are carried out on both synthetic and real datasets to evaluate the performance of SPMLS.

Keywords: Data Mining, Sequential Patterns, Long Sequences

1. INTRODUCTION

Sequential pattern mining plays an important role in the data mining field owing to its wide spectrum of applications. Examples of such applications comprise analysis of web access, customers shopping patterns, stock markets trends, and DNA chains and so on. Initially, it was presented by Agrawal and Srikant. Given a set of sequences, where every sequence has a set of elements. Each element comprises of a list of items, and given a user-specified min support threshold. Sequential pattern mining is used to discover all of the frequent subsequences. The subsequences whose occurrence frequency in the set of sequences is not lesser than min support. The key approach for sequence mining growing from this study is sequence growth.

The sequence growth approach does not need candidate generation. It increasingly raises the frequent sequences. PrefixSpan, which invented from FP-growth, uses this approach as follows. Initially, it discovers the frequent single items. Then it produces a set of projected databases, one database for each frequent item. Each of these databases is then mined recursively while combining the frequent single items into a frequent sequence. These algorithms execute well in databases comprising of short frequent sequences. Though, when mining databases comprising of long frequent sequences, e.g. stocks values, DNA chains or machine monitoring data, their overall performance exacerbates by an order of magnitude.

Integrating constraints in the process of mining sequential patterns is a way to enhance the efficiency of this process and to prevent ineffective and remaining output. cSPADE is an extension of SPADE which efficiently considers a adaptable set of syntactic constraints. These constraints are fully integrated inside the mining process, with no post-processing step. Pei et al. also discuss the problem of pushing various constraints deep into sequential pattern mining. They identify the prefix-monotone property as the common property of

constraints for sequential pattern mining and present a framework (Prefix-growth) that incorporates these constraints into the mining process. Prefix-growth leans on the sequence growth approach.

In this paper we introduce SPMLS, a Sequential Pattern Mining on Long Sequences. SPMLS is developed for high performance on a class of domains categorized by long sequences. Each event is collection of a huge number of items. SPMLS coprises of two phases, event-wise and sequence-wise, which employ an iterative process of candidate-generation followed by frequency-testing. The event-wise phase discovers frequent events satisfying constraints within an event (e.g. two items that cannot reside within the same event). The sequence-wise phase constructs the frequent sequences and enforces constraints between events within these sequences (e.g. two events must occur one after another within a specified time interval). This separation allows the introduction of a novel pruning strategy which reduces the size of the search space considerably.

2. LITERATURE REVIEW

The sequential pattern mining problem [1] introduced by Agrawal and Srikant has been of great interest for researchers as it has a very wide scope of applications spanning from the predicting the customer purchase patterns, and scientific discoveries [2]. For example, given a time stamped sequences of purchases made by customers, where each event is made up of list of items purchased at the same timestamp by a customer. The objective of the sequential pattern mining algorithm is to discover all the frequent sequences. A sequence is called a frequent sequence provided it occurs more frequently than the minimum support specified by users.

There are several heuristics such as GSP[3], SPADE [4], PrefixSpan [5] and the SPIRIT [5] that try to discover the frequent patterns in a efficient manner by trying to prune a large number of subsequences and thus reduce the search space. The GSP algorithm [5] uses the anti-monotone property (all subsequences of a frequent sequence are also frequent). The SPADE finds frequent sequences using the lattice search [6] and intersection based approach. In this approach the sequence database is transformed to a vertical format. The candidate sequences are divided in groups. The frequent sequences are enumerated in SPADE using both methods – breadth first and depth first methods. The support is counted for the generated sequences. The basic approach of the above three algorithm can be classified as the candidate-generation followed by support evaluation. The PrefixSpan [7] algorithm follows a pattern growth method. It uses projected databases for achieving this. Prefix is Projected Sequential Pattern mining. It examines prefix subsequences and projects the postfix subsequences into the databases [8].

Discovering interesting patterns in long sequences is a popular area in data mining [9]. Most existing methods define patterns as interesting if they occur frequently enough in a sufficiently cohesive form. The algorithm proposed in [10] used top-down approach for mining long sequences which defines dominancy of the sequences and uses it for minimizing the scanning of the data set [13]. By performing transformations to the initial dataset, we bring the dominant transactions i.e. those transactions that have longer potentially frequent patterns, to the top in order to mine the long patterns efficiently [14].

3. SEQUENTIAL PATTERN MINING ALGORITHM FOR MINING LONG SEQUENCES

3.1. Definitions

An item is a value assigned to an attribute in our domain. We denote an item as a letter of the alphabet: a, b, ..., z. Let A = $\{a_1, a_2, ...a_m\}$ be the set of all possible items in our domain. An event is a nonempty set of items that occurred at the same time. We denote an event as

$$b = \{a_1, a_2, ... a_n\},$$

where $a_j \in A$, $1 \leq j \leq n$. An event containing l items is called an l-event. For example, (bcd) is a 3-event. If every item of event e_1 is also an item of event e_2 then e_1 is said to be a subevent of e_2, denoted $e_1 \subseteq e_2$. Equivalently, we can say that e2 is a super- event of e1 or e2 contains e1. For simplicity, we denote 1-events without the parentheses. Without the loss of generality we assume that items in an event are ordered lexicographically, and that there is a radix order between different events. Notice that if an event e1 is a proper superset of event e2 then e2 is radix ordered before e1. For example, the event (qr) is radix ordered before the event (pqr), and (qr) (pqr).

A sequence is an ordered list of events, where the order of the events in the sequence is the order of their occurrences. We denote a sequence s as,

$$s = \{b_1, b_2, ... b_k\}$$

where b_j is an event, and b_{j-1} happened before b_j . Notice that an item can occur only once in an event but can occur multiple times in different events in the same sequence. A sequence containing k events is called a

k-sequence, in contrast to the classic definition of k-sequence that refers to any sequence containing k items. For example, {(st)t(prs)} is a 3-sequence.

A sequence $c_1 = \{b_1^1, b_2^1, ..., b_n^1\}$ is a subsequence of sequence $c_2 = \{b_1^2, b_2^2, ..., b_m^2\}$, denoted $c_1 \subseteq c_2$, if there exists a series of integers $1 \leq j_1 < j_2 < .... < j_n \leq m$, such that $b_1^1 \subseteq b_{j1}^2 \wedge b_2^1 \subseteq b_{j2}^2 \wedge ..., \wedge b_n^1 \subseteq b_{j1}^2$. Equivalently we say that c_2 is a super-sequence of c_1 or c_2 contains c_1.

A database of sequences is an unordered set of sequences, where every sequence in the database has a unique identifier called sid. Each event in every sequence is associated with a timestamp which states the time duration that passed from the beginning of the sequence. This means that the first event in every sequence has a timestamp of 0. Since a timestamp is unique for each event in a sequence it is used as an event identifier and called eid. The support or frequency of a sequence s, denoted as sup(s), in a sequence database D is the number of sequences in D that contain s. Given an input integer threshold, called minimum support, or minSup, we say that s is frequent if sup(s) ≥ minSup. The frequent patterns mining task is to find all frequent subsequences in a sequence database for a given minimum support.

3.2. Constraints

Constraints are a means of defining the type of sequences one is looking for. We define two types of constraints: intra-event constraints, which refer to constraints that are not time related (such as values of attributes) and inter-events constraints, which relate to the temporal aspect of the data, i.e. values that can or cannot appear one after the other sequentially. For the purpose of the experiment conducted in this study and in accordance with our domain, we chose to incorporate two inter-event and two intra-events constraints. Intra-event Constraints are Singletons and MaxEventLength. Inter-events Constraints are MaxGap and MaxSequenceLength.

3.3. SPMLS Algorithm

We now present SPMLS, a Sequential Pattern Mining on Long Sequences. The algorithm has two phases, event-wise and sequences-wise. The distinction between the two phases corresponds to the difference between the two types of constraints. As explained below, this design enhances the efficiency of the algorithm and makes it readily extensible for accommodating different constraints. Pseudo code for SPMLS is presented.

Input: minSup, maxGap, maxEventLength, maxSeqLength, D (dataset), A (set of singletons).
Output: F (set of all frequent sequences)

Step 1: Takes an input Dataset
Step 2: {Event-wise phase}
Step3:F1←allFrequentEvents(minSup,maxEventLength,A,D)
Step 4: F ← F1
Step 5: {Sequence-wise phase}
Step 6: for all k such that 2 ≤ k ≤ maxSeqLength and Fk−1 6= Á do
Step7:Ck←candidateGen(Fk−1,maxGap)
Step8:Fk←prune(Fk−1,Ck,minSup,maxGap)
Step 9: F ← F ∪ Fk
Step 10 : Mine Long Frequent Sequence

3.3.1. Event-wise Phase

The input to this phase is the database itself and all of the intra-event constraints. During this phase, the algorithm disregards the sequential nature of the data, and treats the data as a set of events rather than a set of sequences. Since the data is not sequential, applying the intra-event constraints at this phase is very straightforward. The algorithm is similar to the Apriori algorithm for discovering frequent itemsets as presented in [1].

Similarly to Apriori, our algorithm utilizes an iterative approach where frequent (k − 1)-events are used to discover frequent k-events. However, unlike Apriori, we calculate the support of an event by counting the number of sequences that it appears in rather than counting the number of super-events that contain it. This means that the appearance of an event in a sequence increases its support by one, regardless of the number of times that it appears in that sequence.

Denoting the set of frequent k-events by Lk (referred to as frequent k-itemsets in [1]), we begin this phase with a complete database scan in order to find L1. Next, L1 is used to find L2, L2 is used to find L3 and so on, until the resulting set is empty, or we have reached the maximum number of items in an event as defined by MaxEventLength. Another difference between Apriori and this phase lies in the generation process of Lk

from Lk−1.When we join two (k−1)-events to generate a k-event candidate we need to check whether the added item satisfies the rest of the intra-event constraints such as Singletons.

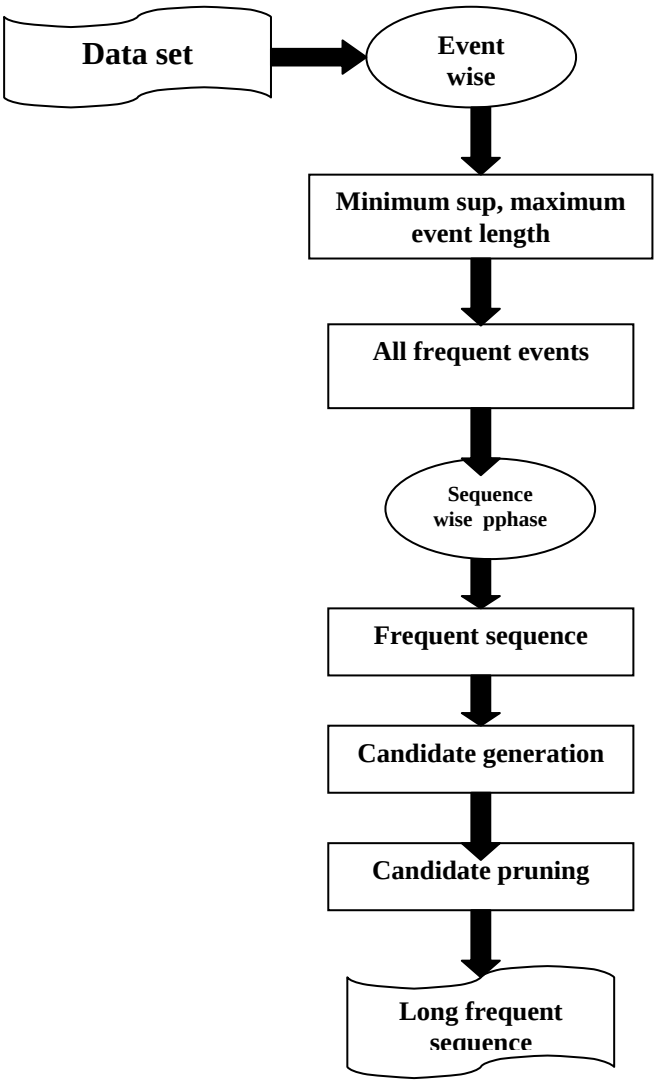


Figure 1: Sequential Pattern Mining on Long Sequences

The output of this phase is a radix ordered set of all frequent events satisfying the intra-event constraints, where every event ei is associated with an occurrence index. The occurrence index of the frequent event ei is a compact representation of all occurrences of ei in the database and is structured as follows: a list li of sids of all sequences in the dataset that contain ei, where every sid is associated with the list of eids of events in this sequence that contain ei. We are able to keep this output in main memory due to the nature of our domain in which the number of frequent events is relatively small. Since the support of ei equals the number of elements in li, it is actually stored in ei's occurrence index. Thus, support is obtained by querying the index instead of scanning the database. In fact, the database scan required to find L1 is actually the only single scan of the database. Throughout the event-wise phase, additional scans are avoided by keeping the occurrence indices for each event.

3.3.2. Sequence-wise Phase

The input to this phase is the output of the previous one. It is entirely temporal- based, therefore applying the inter-events constraints is straightforward. The output of this phase is a list of frequent sequences satisfying the inter-events constraints. Since it builds on frequent events that satisfied the intra-event constraints, this output amounts to the final list of sequences that satisfy the complete set of constraints.

Similarly to SPADE, this phase of the algorithm finds all frequent sequences by employing an iterative candidate generation method based on the apriori property. For the kth iteration, the algorithm outputs the set of frequent k-sequences, denoted as Fk, as follows: it starts by generating the set of all k- sequence candidates,

denoted as Ck from Fk−1 (found in the previous iteration). Then it prunes candidates that do not require support calculation in order to determine that they are non-frequent. Finally, it calculates the remaining candidates' support and removes the non-frequent ones.

Sequential pattern mining requires frequent item sets, the arrangement of those itemsets in frequent sequences. It exists a significant increase in the number of potential patterns, assume that there is a database to be mined with the minimum support threshold set to s and with p=|C| different items in the item collection, C. The goal of frequent itemset mining is to find which itemsets are frequent from the | A | different possible existent itemsets, where A is the power set of C, and its value is given by equation (1).

$$|A| = \sum_{j=1}^{p} \binom{p}{j} - 1 = 2^{p} - 1 \qquad (1)$$

To understand the sequential pattern mining problem, let's begin by considering that the database has sequences with at most q itemsets and each itemset has at most one item. In these conditions, there would be pq possible different sequences with q itemsets (2)

$$\sum_{k=1}^{q} p^{k} = \frac{p^{q+1} - p}{p - 1} \qquad (2)$$

Different arbitrary length sequences. Similarly, if each itemset has an arbitrary number of items, there would exist Cq possible frequent sequences with q itemsets, with the value of cq given by equation (3).

$$c_q = |A|^q = (2^p - 1)^q \qquad (3)$$

And, there would exist C sequences in general, as in equation (4).

$$c = \sum_{k=1}^{q} (2^p - 1)^k = \frac{(2^p - 1)^{q+1} - 2^p - 1}{2^p - 2} = 0(2^{pq}) \qquad (4)$$

Indeed, the number of different items and the average length of frequent sequences are tightly connected: a large number of items in a short sequence may imply a reduced number of frequent patterns, since the probability of the generality of items has to be small. In this way, algorithms will run efficiently. The opposite situation is a large sequence with a reduced number of items, where the probability of each element to occur in a large number of sequences is high. This leads to the existence of many patterns, and consequently a large amount of processing time. The concept of database density quantifies this relationship. The database density is the ratio between the number of existing patterns (F) and the number of possible sequences C.

4. EXPERIMENTAL RESULTS

The experimental evaluation of the proposed algorithm SPMLS, Sequential Pattern Mining on Long Sequences, is conducted on evaluating its performance in Time series dense data sets in arff format in Weka tool compared it with CAMLS. We compared the run-time of the algorithms by applying them on real data sets. We conducted several runs with and without including constraints.

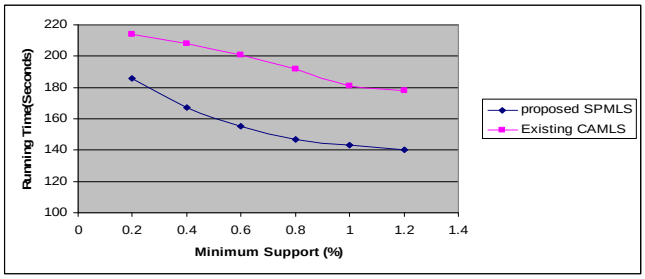
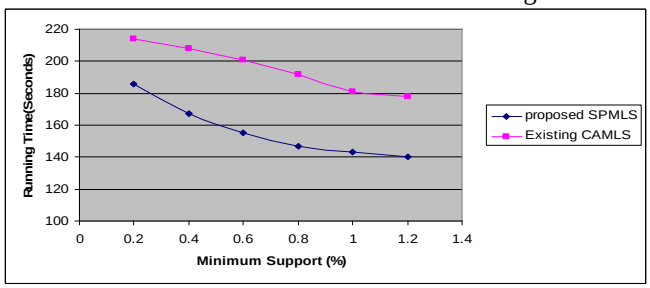


Figure 2: Running Time

Figure 2 compares SPMLS with CAMLS on real datasets. The graph shows the change in execution-time as the minimum support descends from 0.2 to 1.2. The amount of frequent patterns found for each minimum support value is indicated by the number that appears above the respective value. This comparison indicates that SPMLS has a advantage on datasets of long sequences with long e



vents.

Figure 3: Running Time

In the Real dataset, where sequences are especially long, SPMLS clearly outperforms CAMLS for all minimum support values tested.

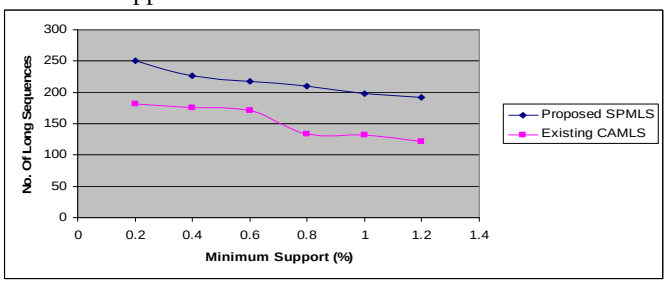


Figure 4: No. Of Long Sequences

Figure 4 compares SPMLS and CAMLS with the usage of the maxGap and Singletons constraints. On real datasets, SPMLS outperforms CAMLS.

5. CONCLUSION

In this paper we proposed SPMLS, a Sequential Pattern Mining on Long Sequences. SPMLS implemented in two phases reflecting a conceptual distinction between the treatment of temporal and non temporal data. Temporal aspects are only relevant during the sequence-wise phase while non temporal aspects are dealt with only in the event-wise phase. There are two primary advantages to this distinction. First, it allows us to apply a novel pruning strategy which accelerates the mining process. The accumulative effect of this strategy becomes especially apparent in the presence of many long frequent sequences. Second, the incorporation of inter-event and intra-event constraints, each in its associated phase, is straightforward and the algorithm can be easily extended to include other inter-events and intra-events constraints. Experimental results compare SPMLS to known algorithms and show that its advantage increases as the mined sequences get longer and the number of frequent patterns in them rises.

References

[1] B. Cule, B. Goethals and C. Robardet, A new constraint for mining sets in sequences , Proc. SIAM Int. Conf. on Data Mining (SDM), pp. 317-328, 2009

[2] N. Mabroukeh and C. Ezeife. Taxonomy of sequential pattern mining algorithms. ACM Computing Surveys Journal, 2009.

[3] Jinlin Chen, Member, IEEE, “An UpDown Directed Acyclic Graph Approach for Sequential Pattern Mining”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 22, NO. 7, JULY 2010.

[4] Faraz Rasheed, Mohammed Alshalalfa Reda and Alhajj,” Efficient Periodicity Mining in Time Series Databases Using Suffix Trees”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING-2011.

[5] David Lo, Member, IEEE, Jinyan Li, Limsoon Wong, and Siau-Cheng Khoo, “Mining Iterative Generators and Representative Rules for Software Specification Discovery”, IEEE Transactions On Knowledge And Data Engineering, VOL. 23, NO. 2, FEBRUARY 2011.

[6] Jae-Gil Lee, Member, IEEE, Jiawei Han, Fellow, IEEE, Xiaolei Li, and Hong Cheng, “Mining Discriminative Patterns for Classifying Trajectories on Road Networks”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 23, NO. 5, MAY 2011.

[7] Hsiao-Ping Tsai, Member, IEEE, De-Nian Yang, and Ming-Syan Chen, Fellow, IEEE,” Mining Group Movement Patterns for Tracking Moving Objects Efficiently”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 23, NO. 2, FEBRUARY 2011.

[8] Mahdi Esmaieli and Mansour Tarafdar, “Sequential Pattern Mining from Multidimensional Sequence Data in Parallel”, International Journal of Computer Theory and Engineering, Vol. 2, No. 5, October, 2010, 1793-8201.

[9] Karam Gouda1 and Mosab Hassaan, "MINING SEQUENTIAL PATTERNS IN DENSE DATABASES",Faculty of Computers and Informatics, Information System Department, Benha University, Egypt, International Journal of Database Management Systems (IJDMS), Vol.3, No.1, February 2011.
[10] David Lo,Siau-Cheng Khoo and Jinyan Li, "Mining and Ranking Generators of Sequential Patterns", Proceedings of SIAM International Conference, 2008.
[11] Mourad Ykhlef, Yousuf Aldukhayyil and Muath Alfawzan, "Mining Sequential Patterns in Telecommunication Database Using Genetic Algorithm",King Saud University, College of Computer and Information Sciences, Information System Department, Kingdom of Saudi Arabia-2008.
[12] Mourad Ykhlef, Assistant Professor, King Saud University, and Hebah ElGibreen, Lecturer, King Saud University, "Mining Sequential Patterns Using Hybrid Evolutionary Algorithm", World Academy of Science, Engineering and Technology 60 2009.
[13] Yong Chen, Rong fang Bie and Chuan Xu, "A New Approach for Maximal Frequent Sequential Patterns Mining Over Data Streams", International Journal of Digital Content Technology and its Applications. Volume 5, Number 6, June 2011.
[14] Jiawei Han, Hong Cheng, Dong Xin and Xifeng Yan, "Frequent pattern mining: current status and future directions", Data Min Knowl Disc (2007) 15:55–86 DOI 10.1007/s10618-006-0059-1.